Quantitative Finance

Hedged Monte-Carlo

Patrick Hénaff

Version: 14 mars 2024

Contents


```
library(lubridate)
library(fExoticOptions)
library(kableExtra)
library(ggplot2)
library(stats)
library(nleqslv)
library(reshape)
```

The principle of the Monte-Carlo in finance is to simulate the diffusion of some underlying asset, and compute the price of a derivative for each simulated path. The expected discounted payoff of the derivative is an estimate of the price of the derivative under consideration.

In contrast, the hedged Monte-Carlo algorithm does not simulate the payoff of the derivative, but the cash-flow (or cashflows) generated by a self-financing dynamic strategy involving the derivative to be priced, and its delta hedge.

The method presents two advantages compared to a standard Monte-Carlo method:

- the simulated dynamic strategy has lower variance that the asset itself, and therefore the price estimate also has lower variance
- by simulating a self-financing strategy, one can price assets under densities that are not risk-neutral. One can, for example, use historical paths to price the derivative.

Those features of the method will be demonstrated at the end of this note.

In this section, we summarize the model developed by Potters *et al* (Potters et al., 2001).

1 The hedged Monte-Carlo algorithm

Let's first introduce some notation:

x_k value of underlying asset at step k
$C_k(x_k)$ value of the derivative
$\phi_k(x_k)$ hedge ratio for derivative C_k

Define the local risk R_k as:

$$E^P\left[(C_{k+1}(x_{k+1}) - C_k(x_k) - \phi_k(x_k)[x_{k+1} - x_k])^2\right]$$

where $E^P[]$ is the expectation under the objective probability measure.

We look for the pricing function $C_k(x)$ that minimizes the residual hedging risk.

The functions $C(x)$ and $\phi(x)$ are approximated by a set of basis functions:

$$C_k(x) = \sum_{a=1}^{M} \gamma_a^k C_a(x) \tag{1}$$

$$\phi_k(x) = \sum_{a=1}^{M} \gamma_a^k \frac{\partial C_a(x)}{\partial x} \tag{2}$$

Splines provide a convenient set of basis functions: given a set of knots $t_i, i = 1 \ldots, k$, the polynomial spline of degree n is defined by:

$$C(x) = \sum_{j=0}^{n} b_{0,j} x^j + \sum_{i=1}^{k} \sum_{j=0}^{n} b_{i,j} (x - t_i)_+^j$$

Thus, a spline of degree n with k knots is a linear combination of $m = (k+1)(n+1)$ basis functions. The derivative of $C(x)$ with respect to x is readily computed. To simplify notation, let:

$$C(x) = \sum_{a=1}^{m} \beta_a F_a(x)$$

$$C'(x) = \sum_{a=1}^{m} \beta_a F_a'(x)$$

where $F_a(x)$ are the elementary basis functions. At each step t in the hedged Monte-Carlo simulation, we obtain the price function by solving for β the following optimization problem (formulation for a call):

$$\min \sum_{l=1}^{N} [e^{-\rho} C_{t+1}(x_{t+1}^l) - \sum_{a=1}^{M} \beta_a (F_a(x_t^l) + F_a'(x_t^l)(x_{t+1}^l e^{-\rho} - x_t^l))]^2$$

A simple modification of the model enables us to account for the bid-ask spread on transactions. We assume that the price paths are mid-market prices, and that transactions are negotiated at the bid or ask price. Let ϵ be the bid-ask spread. The local risk becomes:

$$(C_{k+1}(x_{k+1}) - C_k(x_k) - \phi_k(x_k)(e^{-\rho}x_{k+1} - x_k - \delta\epsilon/2))^2$$

where

$$\delta = \begin{cases} -1 & (x_k - e^{-\rho}x_{k+1}) >= 0 \\ 1 & (x_k - e^{-\rho}x_{k+1}) < 0 \end{cases}$$

So far, we have only considered contracts with a single payoff at expiry. A simple extension of the model can accommodate arbitrary contingent cash flows at each step of the simulation. Assume that as each time step, the contract generates a cash flow $F(x_k)$. We then define the price function $C(x_k)$ as the contract value ex cash flow. The local risk function is then:

$$(C_{k+1}(x_{k+1}) + F_{k+1}(x_{k+1}) - C_k(x_k) + \phi_k(x_k)(x_k - e^{-\rho}x_{k+1} + \delta\epsilon/2))^2$$

With contingent cash flows at each period, the constraints defined by equation (**??**) need to be reformulated. Consider first the constraint that the value of the contract must be greater or equal to the intrinsic value of the European option. With multiple cash flows, the equivalent constraint is that the contract value at a given time step and state must be greater than the sum of the expected discounted cash flows, under the conditional probability of this time step and state. The rest of the algorithm is left unchanged.

2 Illustration

To demonstrate the usefulness of this algorithm, we reproduce some results from (Potters et al., 2001). The first experiment compares the Hedged MC algorithm to the classical binomial model for pricing European options, and to a unhedged Monte-Carlo pricing method. The first step is togenerate geometric brownian paths:

```
GBMPathSimulator <-
  function(nbObs, nbSim, S0, mu, sigma, TTM, center = 'Mean') {
    delta.t <- TTM / nbObs
    if (center == 'Mean') {
      Z <- rnorm(nbObs * nbSim,mean = 0,sd = 1)
      dim(Z) <- c(nbObs, nbSim)
      Z <- Z - mean(Z)
    } else {
      Z <- rnorm(nbObs * nbSim / 2,mean = 0,sd = 1)
      dim(Z) <- c(nbObs, nbSim / 2)
      Z <- cbind(Z,-Z)
    }

    path = (mu - sigma * sigma / 2) * delta.t + sigma * sqrt(delta.t) * Z
    S <- S0 * rbind(rep(1, nbSim), exp(apply(path, 2, cumsum)))
```

```
  S
}
```

For the sake of simplicity, the basis functions are truncated power functions with one knot located at the strike.

```
basis <- function(k, S, K, truncate=FALSE) {
  truncated.S <- if(truncate) pmax(S - K, 0) else S
  F <- truncated.S ^ (k-1)
  F.prime <- (k-1)*truncated.S^(k-2)
  cbind(F, F.prime)
}
```

The hedged Monte-Carlo algorithm is implemented as follows:

```
nb.paths <- 500  # number of simulated paths
M <- 8           # number of splines
X <- matrix(0, nrow=nb.paths, ncol=M+1)
Z <- matrix(0, nrow=nb.paths, ncol=M+1)
american.ex <- FALSE

hedged.mc <- function(payoff, K) {
  # continuation value at last time step is exercice value
  CV <- payoff(S[,(nb.steps+1)])
  t.last=1
  for (t in nb.steps:t.last) {
    if(american.ex) {
      exercice.value <- payoff(S[,t+1])
      CV <- pmax(CV, exercice.value)
    }
    # discounted continuation value
    disc.CV <- CV * df

    # compute all polynomial terms
    for(k in seq(M+1)) {
      tmp <- basis(k, S[,t], K, truncate=FALSE)
      C.alpha <- tmp[,1]
      F.alpha <- tmp[,2]
      X[, k] <- C.alpha + F.alpha * (S[, t+1]*df - S[, t])
      Z[, k] <- C.alpha
    }

    # Try polynomial regression until all coefficients OK

    reg.ok <- FALSE
    MM <- M
    MM.min <- if(t>1) 2 else 1
```

```
    while(!reg.ok & MM>=MM.min) {
      # add truncated term
      tmp <- basis(MM, S[,t], K, truncate=TRUE)
      C.alpha <- tmp[,1]
      F.alpha <- tmp[,2]
      X[, MM+1] <- C.alpha + F.alpha * (S[, t+1]*df - S[, t])
      Z[, MM+1] <- C.alpha

      # at the origin, the independent variable is the constant S.0
      if(t>1) {
        reg <- lm(disc.CV ~ X[,seq(MM+1)] -1, singular.ok = TRUE)
      } else {
        reg <- lm(disc.CV ~ X[,seq(MM)] -1, singular.ok = TRUE)
      }
      reg.ok <- (any(is.na(reg$coefficients)) == FALSE)
      if(!reg.ok) MM <- MM-1
    }
    if(!reg.ok) {
      stop(paste("Regression error at t:", t))
    }

    if(t>1) {
      CV <- Z[,seq(MM+1)] %*% matrix(reg$coefficients, ncol=1)
    } else {
      CV <- Z[,seq(MM)] %*% matrix(reg$coefficients, ncol=1)
    }
  }
 mean(CV)
}
```

The first experiment reported in Potters's paper involves a 3-months Call option, strike 100.

```
  K.call <- 100
  call.payoff <- function(S) pmax(S-K.call, 0)
```

The other parameters of the experiment are as follows:

```
S.0 <- 100    # spot
r <- 0.05     # risk-free interest rate
div <- 0      # dividend yield
TTM <- 1/4    # time to maturity, in years
sigma <- 0.30 # volatility

nb.steps <- 20   # number of time steps in simulation
nb.trials <- 500 # number of replications

dT <- TTM/nb.steps
```

Table 1: Pricing an ATM European option maturity 3 months. $S_0 = 100$

	Black-Scholes	Unhedged MC	Hedged MC
Mean	6.58	6.58	6.57
SD	NA	0.24	0.06

```
# discount factor for one time step
df <- exp(-r*dT)
```

We compare in Table 1 the exact Black-Scholes value of the option to the estimates obtained by unhedged Monte-Carlo simulations and by hedged simulations.

```
hedged.mc.price <- vector(mode="numeric", length=nb.trials)
unhedged.mc.price <- vector(mode="numeric", length=nb.trials)

for(i in seq(nb.trials)) {
  # simulated paths for the underlying asset
  S <- t(GBMPathSimulator(nb.steps, nb.paths, S.0, mu=r-div,

  unhedged.mc.price[i] <- mean(call.payoff(S[, nb.steps+1])) * exp(-r*TTM)
  hedged.mc.price[i] <- hedged.mc(call.payoff, K.call)
}

opt <- CRRBinomialTreeOption(TypeFlag="ce", S=S.0, X=K.call, Time=TTM, r=r,
                             b=r, sigma=sigma, n=200)@price
```

The results show a remarkable consistency between the Hedged MC method and the Black-Scholes value, with a standard deviation reduced by x% with respect to the unhedged MC algorithm.

The true value of the Hedged MC algorithm is its ability to price an asset under a density that is not risk-neutral. This is demonstrated by the next experiment, where the drift of the geometric brownian motion is set to 30%. Everything else is left as in the previous experiment:

```
drift <- 0.3
for(i in seq(nb.trials)) {
  # simulated paths for the underlying asset
  S <- t(GBMPathSimulator(nb.steps, nb.paths, S.0, mu=drift-div,

  unhedged.mc.price[i] <- mean(call.payoff(S[, nb.steps+1])) * exp(-r*TTM)
  hedged.mc.price[i] <- hedged.mc(call.payoff, K.call)
}
```

The results in Table 2 show that the Hedged MC correctly prices the option under a geometric brownian motion with an arbitrary drift.

In order to evaluate an American put option, the estimated value $C_{k+1}(x_{k+1})$ is replaced by $\max(C_{k+1}(x_{k+1}), K - x_{k+1})$, where K is the strike.

Table 2: Pricing an ATM European option when the scenarios are not risk-neutral. $S_0 = 100$

	Black-Scholes	Unhedged MC	Hedged MC
Mean	6.58	10.72	6.54
SD	NA	0.23	0.07

Table 3: Pricing an ATM American option, maturity 1 year. $S_0 = 40$

	Binomial	Hedged MC
Mean	2.32	2.35
SD	NA	0.03

Following Potters's paper, the other parameters are as follows:

```
S.0 <- 40      # spot
r <- 0.06      # risk-free interest rate
div <- 0       # dividend yield
TTM <- 1       # time to maturity, in years
sigma <- 0.20 # volatility

nb.steps <- 20    # number of time steps in simulation
nb.paths <- 500   # number of simulated paths
nb.trials <- 500 # number of replications

K.put <- 40
put.payoff <- function(S) pmax(K.put-S, 0)

dT <- TTM/nb.steps
# discount factor for one time step
df <- exp(-r*dT)
```

```
american.ex <- TRUE
for(i in seq(nb.trials)) {
  # simulated paths for the underlying asset
  S <- t(GBMPathSimulator(nb.steps, nb.paths, S.0, mu=r-div,

  hedged.mc.price[i] <- hedged.mc(put.payoff, K.put)
}

opt <- CRRBinomialTreeOption(TypeFlag="pa", S=S.0, X=K.put, Time=TTM, r=r,
                             b=r, sigma=sigma, n=200)@price
```

Table 3 summarizes the results. The estimated price by hedged MC is higher than reported, but qualitatively consistent with the binomial price.

References

Potters, M., Bouchaud, J., & Sestovic, D. (2001). Hedged monte-Carlo: low variance derivative pricing with objective probabilities. *Physica*, *289*, 517–525.