

# Quantitative Finance

Monte-Carlo

Patrick Hénaff

Version: 10 Mar 2025

## Monte Carlo Methods

There are many instances in computational finance where one needs to compute the expected value of some function of a multivariate random variable. One can think about an option on a basket of stocks, or an option where the payoff is determined by observations performed at regular intervals in time, and so forth.

Monte Carlo methods, in contrast to lattice methods such as trees, become particularly attractive as the dimension of the domain increases. This can be illustrated by considering the problem of evaluating a volume  $V$  in  $R^d$ , by evaluating a function  $f(X)$  at  $N$  points such that  $N = k^d$  for some integer  $k$ . Along each axis, the  $k$  points are evenly spaced. If a mid-point integration algorithm is used, then the error is of the order of  $1/k$ , and  $1/k^2$  if a trapezoidal rule is used. Therefore, with  $N$  function evaluations, the error will be  $O(N^{-1/d})$  or  $O(N^{-2/d})$ , depending upon the integration rule.

In contrast, consider performing the same calculation by evaluating  $g(X)$  at  $N$  random points such that  $g(X) = 1$  if  $X \in V$  and 0 otherwise. The estimate  $S_N$  of the volume is

$$S_N = C_d \frac{1}{N} \sum_{i=1}^N f(X_i)$$

where  $C_d$  is the volume of the hypercube of dimension  $d$  containing the sphere. An estimation of the variance of the estimate is

$$V_n^2 = \frac{N-1}{N} \left[ \frac{1}{N} \sum_i g(X_i)^2 - \left( \frac{1}{N} \sum_i g(X_i) \right)^2 \right]$$

which shows, by the central limit theorem, that the error is a function of  $\sqrt{1/N}$ , and not of the dimension of the domain.

$$S_n - E(g(X)) \approx N(0, \sqrt{1/N} V_n)$$

The following experiments contrast the lattice and Monte-Carlo methods. We estimate the volume of a sphere of radius 1 in  $R^d$  contained in an hypercube of the same dimension. The volume of the sphere is estimated by considering  $N$  points in  $R^d$  and determining if they are inside the sphere. Let  $K$  be the number of such points, the volume of the sphere is estimated as:

$$V_N = \frac{K}{N} C_d$$

The lattice and Monte-Carlo methods differ in the way the  $N$  points are chosen. With the lattice algorithm, we construct a grid of  $N$  evenly spaced points in  $R^d$ , while in the Monte-Carlo approach, those  $N$  points are drawn from a uniform multivariate distribution. Since we have an analytical expression for the volume of the sphere, we can readily observe the estimation error as a function of  $N$  and the dimension  $d$ .

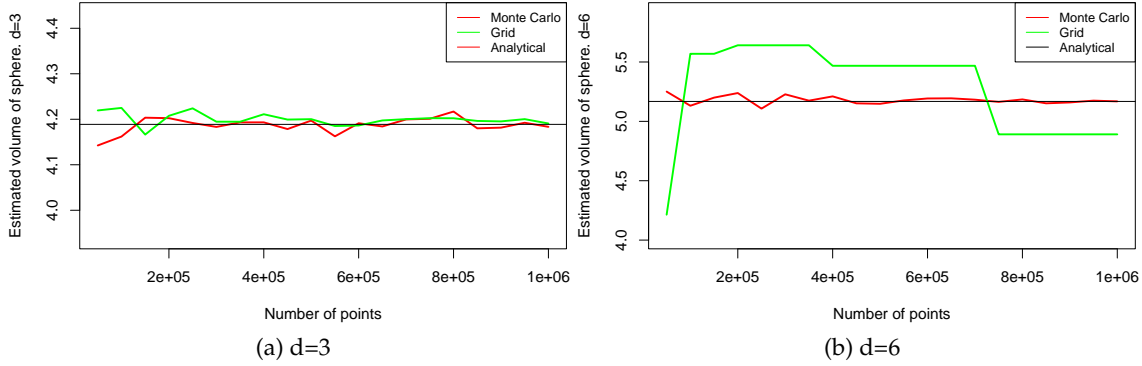


Figure 1: Calculation of the volume of a sphere in dimension 3 and 6, with an evenly spaced grid and Monte-Carlo simulation.

For  $d = 3$ , the difference between the two methods is minimal. In 6 dimensions, however, the superiority of the Monte Carlo algorithm becomes clearly visible. With  $10^6$  simulations, the Monte-Carlo error is  $O(1/\sqrt{N}) = 0.001$ , but  $O(N^{-1/6}) = .1$  with the deterministic grid method. In 6 dimensions, with one million points, there are only 10 grid points per axis.

Another reason makes Monte-Carlo methods attractive in financial engineering, independently of the mathematical properties of the algorithm: a Monte-Carlo simulator, combined with a payoff language, forms a very flexible platform for pricing complex derivatives, without the need for ad-hoc software development. In addition, it provides the framework for easily testing the sensitivity of prices to assumptions regarding the underlying stochastic processes. All these reasons make a Monte-Carlo pricing framework the tool of choice for structured products pricing and risk management.

## Process-driven sampling

In computational finance, we typically need to simulate a stochastic process and evaluate a pricing function for each simulated path. In general terms, the stochastic differential equation to be discretized is

$$dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW$$

Integrating between  $t$  and  $t + \Delta t$  to get  $S_{t+\Delta t}$  gives:

$$S_{t+\Delta t} = S_t + \int_t^{t+\Delta t} \mu(S_u, u)du + \int_t^{t+\Delta t} \sigma(S_u, u)dW_u \quad (\#eq : mc - 1)$$

Various schemes are available to discretize the integrals in @ref(eq:mc-1), Two popular schemes, the Euler and Milstein schemes, are briefly compared.

The Euler scheme is the simplest way to approximate @ref(eq:mc-1): it amounts to approximate the integral using the left-point rule (since  $\mu(S_t, t)$  and  $\sigma(S_t, t)$  are known at time  $t$ ). Nothing that

$$\int_t^{t+\Delta t} \sigma(S_u, u)dW_u \approx \sigma(S_t, t) \int_t^{t+\Delta t} dW_u \quad (1)$$

$$\approx \sigma(S_t, t) (W_{t+\Delta t} - W_t) \quad (2)$$

$$\approx \sigma(S_t, t) \sqrt{\Delta t} Z \quad (3)$$

with  $Z \sim N(0, 1)$ .

The Euler scheme is thus:

$$S_{t+\Delta t} = S_t + \mu(S_t, t)\Delta t + \sigma(S_t, t)\sqrt{\Delta t}Z \quad (\#eq : mc - 2)$$

For the geometric brownian motion

$$dS_t = rS_t dt + \sigma S_t dW \quad (\#eq : mc - gbm)$$

applying @ref(eq:mc-2) gives

$$S_{t+\Delta t} = S_t + rS_t \Delta t + \sigma S_t \sqrt{\Delta t} Z$$

Using Ito's lemma with  $f(S_t) = \ln(S_t)$ , we obtain

$$d \ln(S_t) = (r - \frac{1}{2}\sigma^2)dt + \sigma dW \quad (\#eq : mc - 3)$$

and the corresponding Euler scheme is

$$\ln(S_{t+\Delta t}) = \ln(S_t) + (r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z$$

Milstein's scheme will be considered in the case where  $\mu$  and  $\sigma$  are only functions of  $S_t$ .

$$dS_t = \mu(S_t)dt + \sigma(S_t)dW$$

The accuracy of the discretization is improved by expanding  $\mu(S_t)$  and  $\sigma(S_t)$  using Ito's lemma. The resulting discretization scheme is

$$S_{t+\Delta t} = S_t + \mu_t dt + \sigma_t \sqrt{\Delta t} Z + \frac{1}{2} \sigma'_t \sigma_t \Delta t (Z^2 - 1)$$

Applied to @ref(eq:mc-gbm), the Milstein scheme add a corrective term to the discretization:

$$S_{t+\Delta t} = S_t + r S_t \Delta t + \sigma \sqrt{\Delta t} Z + \frac{1}{2} \sigma^2 S_t \Delta t (Z^2 - 1)$$

However, applied to @ref(eq:mc-3), with  $\mu(S_t) = r - \frac{\sigma^2}{2}$  and  $\sigma(S_t) = \sigma$ , the scheme becomes

$$\ln(S_{t+\Delta t}) = \ln(S_t) + (r - \frac{1}{2} \sigma^2) dt + \sigma \sqrt{\Delta t} Z \text{ (eq : mc - 4)}$$

which is identical to the Euler scheme. It is therefore common practice to use the discretization of  $\ln(S_t)$  provided by @ref(eq:mc-4).

The accuracy of the scheme can be investigated by testing the non-arbitrage condition

$$E(S_T) = S_0 e^{rT}$$

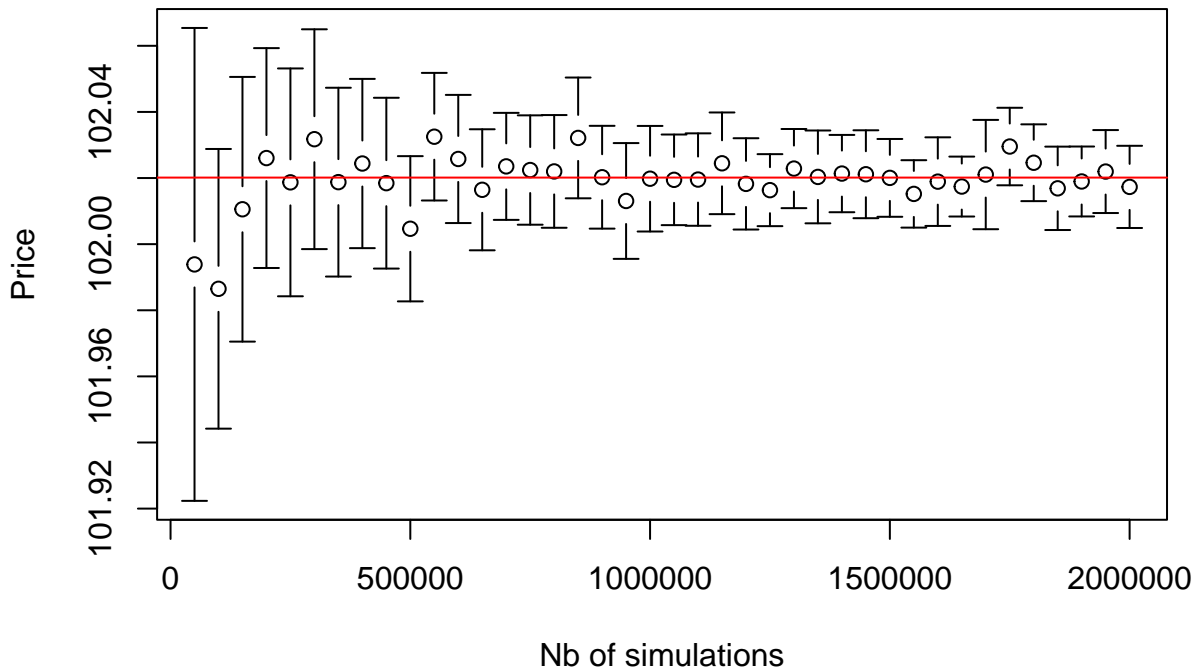


Figure 2: Expected value and s.d. of  $S(T)$  as a function of the number of simulations

Figure @ref(fig:mc-forward) shows the slow progress of  $E(S_T)$  towards the actual forward as the number of simulations increases. Clearly, a naive implementation of the discretization scheme is not satisfactory, and a variance reduction strategy is in order.

## Variance Reduction

Antithetic variates is the simplest improvement to the discretization scheme. For an estimation with  $N$  trials, one draws  $N/2$  variates  $Z_i, i = 1, \dots, \frac{N}{2}$  from  $N(0, 1)$ , and sets  $Z_{N/2+i} = -Z_i, i = N/2 + 1, \dots, N$ . This has the double advantage of ensuring that  $E(Z)$  is exactly null, irrespective of  $N$ , and also reduces the variance of the estimate. Consider the estimation of a parameter  $\theta = E(f(X))$  by Monte-Carlo simulation:

$$\hat{\theta} = \frac{1}{N} \sum_i f(X_i)$$

where the  $X_i$  are sampled according to the distribution of  $X$ . With antithetic variates, the estimate is given by:

$$\hat{\theta}_1 = \frac{2}{N} \sum_{i=1}^{N/2} f(X_i) \quad (4)$$

$$\hat{\theta}_2 = \frac{2}{N} \sum_{i=1}^{N/2} f(-X_i) \quad (5)$$

$$\hat{\theta} = \frac{\theta_1 + \theta_2}{2} \quad (6)$$

and the variance of  $\hat{\theta}$  is

$$V(\hat{\theta}) = \frac{1}{4} [V(\hat{\theta}_1) + V(\hat{\theta}_2) + 2Cov(\theta_1, \theta_2)] \quad (7)$$

$$= \frac{\sigma^2}{N} + \frac{1}{2}Cov(\theta_1, \theta_2) \quad (8)$$

where  $\sigma^2$  is the variance of  $f(X)$ . The scheme will be beneficial as long as  $Cov(\theta_1, \theta_2) < 0$ .

We apply this technique to the pricing of an ATM European call, strike  $K = 100$  and maturity  $T = 1$ , and compare the standard error of the price estimate, with and without antithetic trials. The results are summarized in Figure @ref(fig:mc-call-anti). The antithetic scheme provides at best a moderate improvement.

### Quasi Random Sequences

The discussion of algorithms for generating random or quasi-random numbers is outside the scope of these notes. It will be sufficient to remark that the use of quasi-random numbers such as Sobol sequences significantly improve the quality of the MC estimations. A comparison of Sobol quasi-random numbers and the default uniform random sequence in two dimensions is shown in Figure @fig:rand). The Sobol variates are evenly distributed over the domain.

To appreciate the effect of a Sobol random number generator on the outcome of a MC simulation, we again estimate the forward price  $E(S_T)$ , and compare the calculations performed with a Sobol sequence and with the other schemes considered so far. On these tests, the superiority of the Sobol

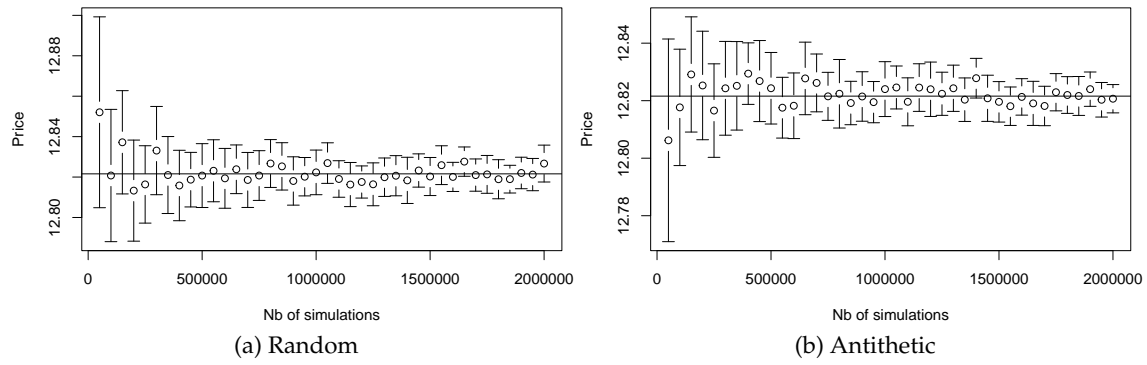


Figure 3: European call with and without antithetic variates

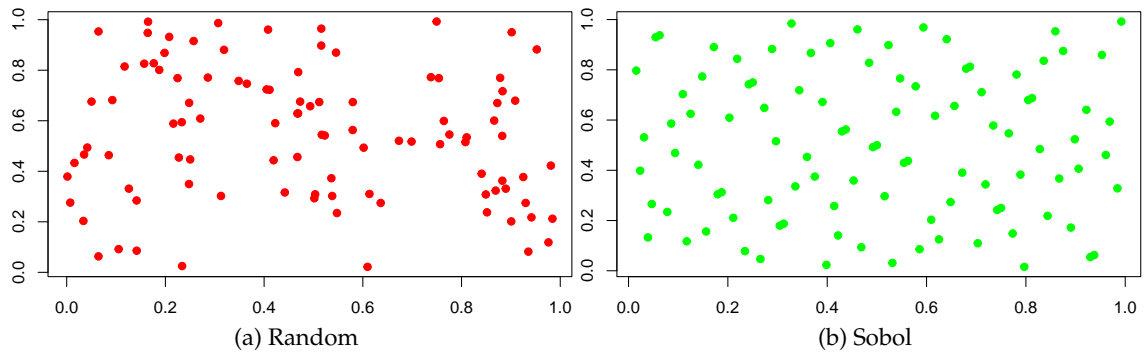


Figure 4: Random variables

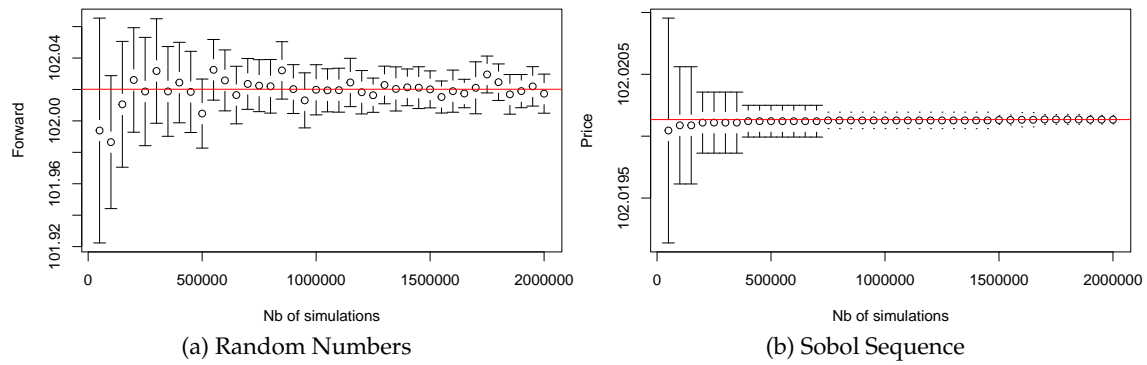


Figure 5: Expected value of  $S(T)$

scheme is apparent. Note that it does not make sense to construct antithetic variates from Sobol sequences, as this would disrupt the even distribution of the variates over the domain.

### Importance Sampling

The principle of importance sampling is to focus on the scenarios that contribute to the average of the Monte-Carlo simulations, for instance, on the scenarios where an option expires in the money.

Pricing a financial asset amounts to computing

$$P = \int_X p(X)\phi(X)dX$$

where  $\phi(X)$  is the payoff, function of the state variables  $X$ , and  $p(X)$  is the density of  $X$ . Assume that  $\phi(X)$  only takes values in a subset  $A$  of the domain of  $X$ , and let  $h(X)$  be the density of  $X$ , restricted to the subset  $A$ , we have:

$$P = \int_{X \in A} p(X)\phi(X)dX \tag{9}$$

$$= \int_{X \in A} p(X) \frac{h(X)}{h(X)} \phi(X)dX \tag{10}$$

$$= \int_{X \in A} h(X) \frac{p(X)}{h(X)} \phi(X)dX \tag{11}$$

$$= E_h \left[ \frac{p(X)}{h(X)} \phi(X) \right] \tag{12}$$

Using  $N$  trials, with  $N_A$  scenarios in subset  $A$ , the value  $P$  is estimated by:

$$P \approx \frac{1}{N_A} \sum_{i=1}^{N_A} \frac{p(X_i)}{h(X_i)} \phi(X_i)$$

Let's apply this algorithm to the pricing of an European call option with strike  $K$  and maturity  $T$  by Monte-Carlo simulation with  $N$  scenarios. The subset  $A$  consists of the  $N_A$  scenarios where  $S_T > K$ . In this simple case,  $p(X) = 1/N$ ,  $h(X) = 1/N_A$  and the estimated call price, ignoring discounting, is

$$P \approx \frac{1}{N} \sum_{i=1}^{N_A} (S_T^i - K)$$

where  $S_T^i, i = 1, \dots, N_A$  are the scenarios ending in the money.

The standard error of this estimate is

$$sd(P) = \frac{\sqrt{N_A}}{N} sd(S_T)$$

The effectiveness of this variance reduction technique is particularly clear when pricing out of the money options, as the following experiment illustrate.

```

K <- 130
S.0 <- 102
T <- 0.5
sigma <- .2
r <- .02

price.call <- function(nb.sim) {
  Z <- rnorm(nb.sim, mean=0, sd=1)
  S.T = S.0*exp((r - 0.5*sigma^2)*T + sqrt(T)*sigma*Z)

  A.index <- S.T > K
  nb.A <- sum(A.index)
  call.payoff <- pmax(S.T-K, 0)[A.index]
  P.call.is <- exp(-r*T) * mean(call.payoff) * nb.A / nb.sim
  sd.call.is <- exp(-r*T) * sqrt(nb.A) * sd(call.payoff) / nb.sim

  call.payoff <- pmax(S.T-K, 0)
  P.call <- exp(-r*T) * mean(call.payoff)
  sd.call <- sd(call.payoff)/sqrt(nb.sim)
  c(P.call.is, sd.call.is, P.call, sd.call)
}

increment <- 50000
nb.calcs <- 20
sim.range <- increment*seq(nb.calcs)
P.call <- foreach(i=1:nb.calcs, .combine=rbind) %dopar% {
  nb.sim <- i*increment
  price.call(nb.sim)
}
P.BS <- GBSOption(TypeFlag="c", S=S.0, X=K, Time=T, r=r, b=r, sigma=sigma)@price

```

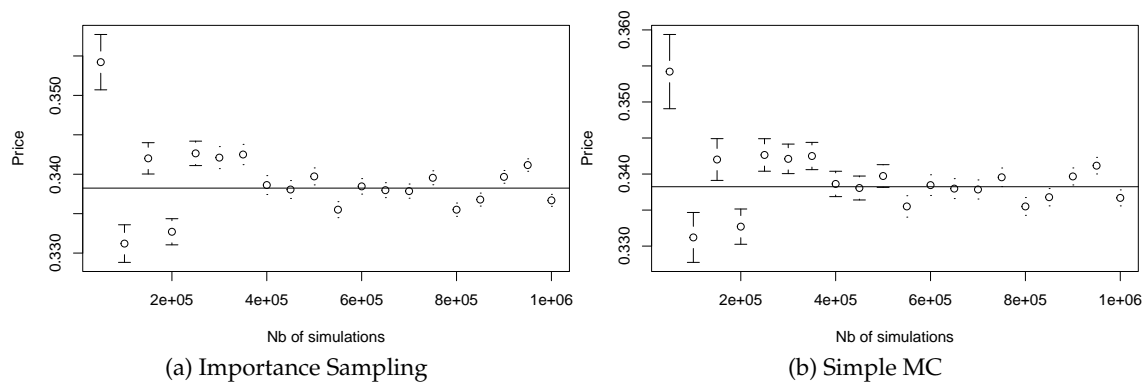


Figure 6: Pricing an European Option

Control Variates



In many instances, we use MC simulations to price a complex product, and we are aware of simpler, related products for which there is an analytical solution. That information can be useful to reduce the variance of our Monte-Carlo simulation.

Let  $g(X)$  be the payoff of the complex instrument to be priced by MC simulation:

$$P_g = \frac{1}{N} \sum_i^n g(X_i)$$

Let  $f(X)$  be the payoff of another instrument for which the expected value  $f^* = E(f)$  is known exactly. We can write:

$$E \left[ \frac{1}{N} \sum_i g(X_i) \right] = E \left[ \frac{1}{N} \sum_i g(X_i) + \beta \left( f^* - \frac{1}{N} \sum_i f(X_i) \right) \right]$$

and the estimator of  $P_g$  with the control variate correction is

$$\hat{P}_g = \frac{1}{N} \sum_i [g(X_i) + \beta(f^* - f(X_i))]$$

with the optimal  $\beta$  being

$$\beta^* = \frac{Cov(f, g)}{V(g)}$$

We illustrate the concept by pricing an Asian option (arithmetic average) and using a European option as control variate.

We create 5000 scenarios, with 50 observations per year.

```
dtExpiry <- dmy('01-01-2011')
dtStart <- dmy('01-01-2010')
nbSteps <- 50
nbPaths <- 100000

S.0 <- 100
sigma=.3
r=.01
T = as.numeric(dtExpiry-dtStart)/365

dtSim <- seq(as.timeDate(dtStart), as.timeDate(dtExpiry),
            length.out=nbSteps+1)

tSpot <- pathSimulator(dtSim = dtSim, nbPaths=nbPaths,
                      innovations.gen=sobolInnovations, path.gen=logNormal,
                      path.param = list(mu=r, sigma=sigma), S0=S.0,
                      antithetic = T,
                      standardization = T, trace = F)
```

Next, we price an European call, strike 100, and compare the estimated price to the analytical value.

```
K <- 100
EuroCall <- function(x) {
  max(x[length(x)]-K,0)
}

payoff.euro <- apply(tSpot,2, FUN=EuroCall)*exp(-r*T)
E.euro <- mean(payoff.euro)
V.euro <- var(payoff.euro)

A.euro <- GBSOption(TypeFlag = "c", S=S.0, X = K, Time = T, r = r,
  b = r, sigma = sigma)@price

print(paste('European Call MC:', round(E.euro,3), '+/-',
  round(sqrt(V.euro/nbPaths),3),
  'BS:', round(A.euro,3)))
```

```
## [1] "European Call MC: 12.431 +/- 0.068 BS: 12.368"
```

We next price a geometric Asian option and compare the estimate to an analytical expression

```
GeomAsianCall <- function(x) {
  avg <- exp(mean(log(x)))
  max(avg-K, 0)
}

payoff.G.asian <- apply(tSpot, 2, FUN=GeomAsianCall) * exp(-r*T)
E.G.asian <- mean(payoff.G.asian)
V.G.asian <- var(payoff.G.asian)
se.G.asian <- sd(payoff.G.asian)/sqrt(nbPaths)

G.exact = GeometricAverageRateOption(TypeFlag="c", S=S.0, X=K, Time=T,
  r=r, b=r, sigma=sigma)@price
print(paste('European Geometric Asian Call MC:', round(E.G.asian,3),
  '+/-', round(se.G.asian,3), 'Exact:', round(G.exact,3)))
```

```
## [1] "European Geometric Asian Call MC: 6.698 +/- 0.035 Exact: 6.701"
```

We then price the Asian option by simulation.

```
EuroAsianCall <- function(x) {
  max(mean(x)-K,0)
}
```

```

payoff.asian <- apply(tSpot, 2, FUN=EuroAsianCall) * exp(-r*T)
E.asian <- mean(payoff.asian)
V.asian <- var(payoff.asian)
se.asian <- sd(payoff.asian)/sqrt(nbPaths)

print(paste('European Asian Call MC:', round(E.asian,3),
            '+/-', round(se.asian,3)))

```

```
## [1] "European Asian Call MC: 7.108 +/- 0.037"
```

Finally, we correct the Monte-Carlo estimate with the control variates, by first estimating the optimal  $\beta$  coefficient.

```

beta <- cov(payoff.asian, payoff.G.asian) / var(payoff.G.asian)
payoff.asian.cv <- payoff.asian + beta*(G.exact-payoff.G.asian)

E.asian.cv <- mean(payoff.asian.cv)
se.asian.cv <- stdev(payoff.asian.cv)/sqrt(nbPaths)
print(paste('Asian price with CV: ', round(E.asian.cv,3),
            ' SE: ', round(se.asian.cv,3),
            ' beta: ', round(beta,2)))

```

```
## [1] "Asian price with CV: 7.111 SE: 0.001 beta: 1.06"
```

```

# Verification
TW = TurnbullWakemanAsianApproxOption(TypeFlag = "c", S = S.0,
    SA = S.0,
    X = K, Time = T, time=1, tau = 0, r = r, b = r,
    sigma = sigma)@price
print(paste("Asian Approx Analytical:", TW))

```

```
## [1] "Asian Approx Analytical: 7.135148729799"
```