P. Henaff

Monte-Carlo Methods

with R and Rmetrics

v 2.0

An Open Access Publication

Contents

1 Monte Carlo Methods 5

1 Monte Carlo Methods

```
library(randtoolbox)
library(foreach)
library(doParallel)
library(gplots)
library(fOptions)
library(fInstrument)
library(DynamicSimulation)
library(empfin)
library(derivmkts)
library(latex2exp)
library(dplyr)
library(tidyr)
library(ggplot2)
library(fExoticOptions)
library(patchwork)
library(gridExtra)
```

There are many instances in computational finance where one needs to compute the expected value of a function of a multivariate random variable. A natural example is the price of an option on a basket of stocks, or the price of an option where the payoff is determined by observations performed at regular intervals in time.

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. A first method is perform a numerical integration, with k points evenly spaced along each axis. 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 an estimation of the variance of the estimate is

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

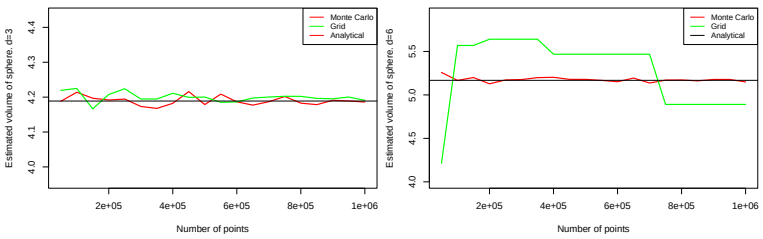
which shows, by the central limit theorem, that the error is a function of $\sqrt{1/N}$, and not of the dimension of the domain.

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

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:

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

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.



(a) d=3 (b) d=6

Figure 1.1: Calculation of the volume of a sphere in dimension 3 and 6, with an evenly spaced grid and by 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 asumptions 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.

1.1 *Process-driven sampling*

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:

$$S_{t+\Delta t} = S_t + \int_t^{t+\Delta t} \mu(S_u, u)du + \int_T^{t+\Delta t} \sigma(S_u, u)dW_u \tag{1.1}$$

Various schemes are available to discretize the integrals in (1.1), Two popular schemes, the Euler and Milstein schemes, are next briefly compared.

The Euler scheme is the simplest way to approximate (1.1): 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

$$\begin{aligned}\int_t^{t+\Delta t} \sigma(S-u, u)dW_u &\approx \sigma(S_t, t) \int dW_u \\ &\approx \sigma(S_t, t)\left(W_{t+\Delta t} - W_t\right) \\ &\approx \sigma(S_t, t)\sqrt{\Delta t}Z\end{aligned}$$

with $Z \sim N(0, 1)$.
The Euler scheme is thus:

$$S_{t+\Delta t} = S_t + \mu(S_t, t)\Delta t + \sigma(S_t, t)\sqrt{\Delta t}Z \tag{1.2}$$

For the geometric brownian motion

$$dS_t = rS_t dt + \sigma S_t dW \tag{1.3}$$

applying (1.2) gives

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

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

$$d\ln(S_t) = (r - \frac{1}{2}\sigma^2)dt + \sigma dW \tag{1.4}$$

and the corresponding Euler scheme is

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

MILSTEIN'S SCHEME will be considered in the case where μ and σ 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 (**??**), the Milstein scheme add a corrective term to the discretization:

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

However, applied to (1.4), with $\mu(S_t) = r - \frac{\sigma^2}{2}$ and $\sigma(S_t) = \sigma$, the scheme becomes

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

which is identical to the Euler scheme. It is therefore common practice to use the discretization of $\ln(S_t)$ provided by (1.1).

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

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

Figure 1.2 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.

1.2 *Variance Reduction*

ANTITHETIC VARIATES is the simplest improvement to the discretization scheme. For an estimation with N trials involving a random variable Z whose density is symmetric with respect to its mean, one draws $N/2$ variates $Z_i, i = 1, \ldots, \frac{N}{2}$, and sets $Z_{N/2+i} = -Z_i, i = N/2+1, \ldots, N$. This has the double advantage of ensuring that $E(Z)$ is exactly null, irrespective of N, and also reduces the variance of the

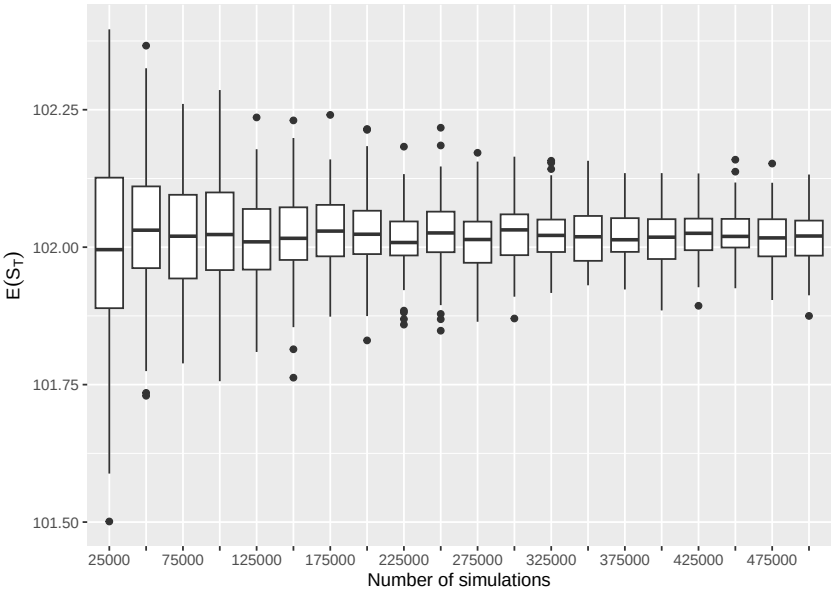


Figure 1.2: Expected value of S_T as a function of the number of simulations The exact value is 102.02.

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:

$$\hat{\theta_1} = \frac{2}{N}\sum_{i=1}^{N/2} f(X_i) \tag{1.5}$$

$$\hat{\theta_2} = \frac{2}{N}\sum_{i=1}^{N/2} f(-X_i) \tag{1.6}$$

$$\hat{\theta} = \frac{\theta_1 + \theta_2}{2} \tag{1.7}$$

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

$$V(\hat{(\theta)}) = \frac{1}{4}\left[V(\hat{\theta_1}) + V(\hat{\theta_2}) + 2Cov(\theta_1, \theta_2)\right] \tag{1.8}$$

$$= \frac{\sigma^2}{N} + \frac{1}{2}Cov(\theta_1, \theta_2) \tag{1.9}$$

where σ^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

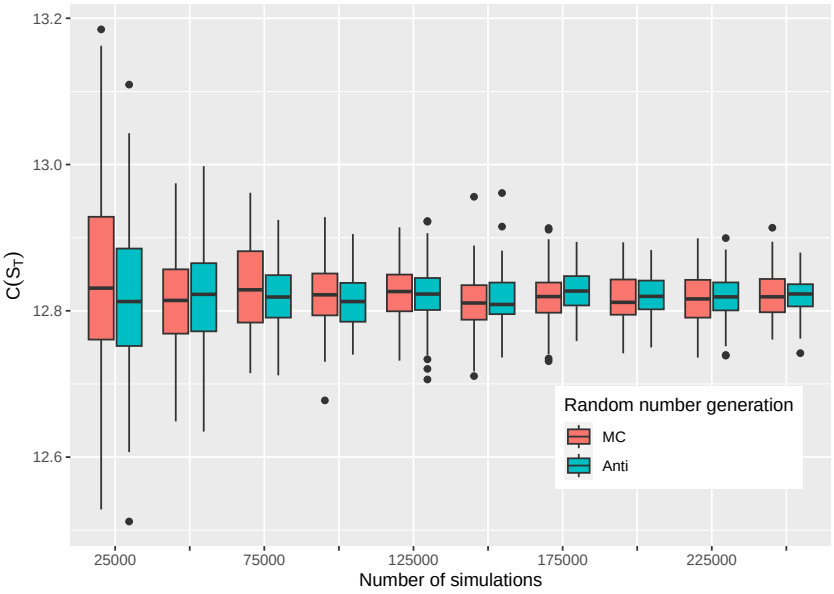
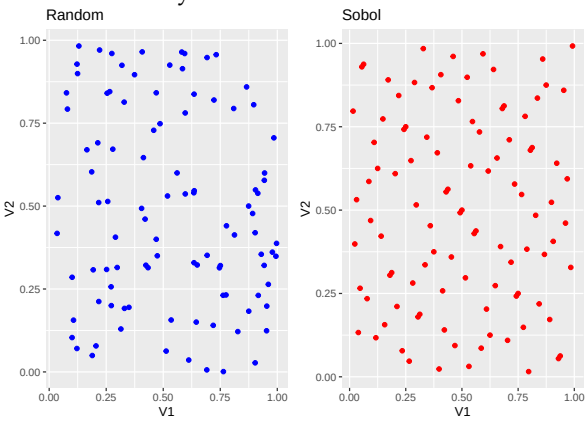


Figure 1.3: Value of a European ATM call as a function of the number of simulations, using simple MC simulations (MC) and antithetic variates (ANTI). The mean and s.d. for each number of simulations is computed with 100 trials. The exact value of the option is ‘r round(P.BS,2)‘

error of the price estimate, with and without antithetic trials. The results are summarized in Figure 1.3. Using antitethic variates yields a reduction of 15.7 % in the standard deviation of the price estimate.

Quasi Random Sequences fill the space with evenly spaced points. The discussion of algorithms for generating such sequences 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 standard MC simulation. On these tests (Figure **??**), the superiority of the antithetic 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. At this stage, using Sobol sequences is by far the most effective variance reduction scheme.

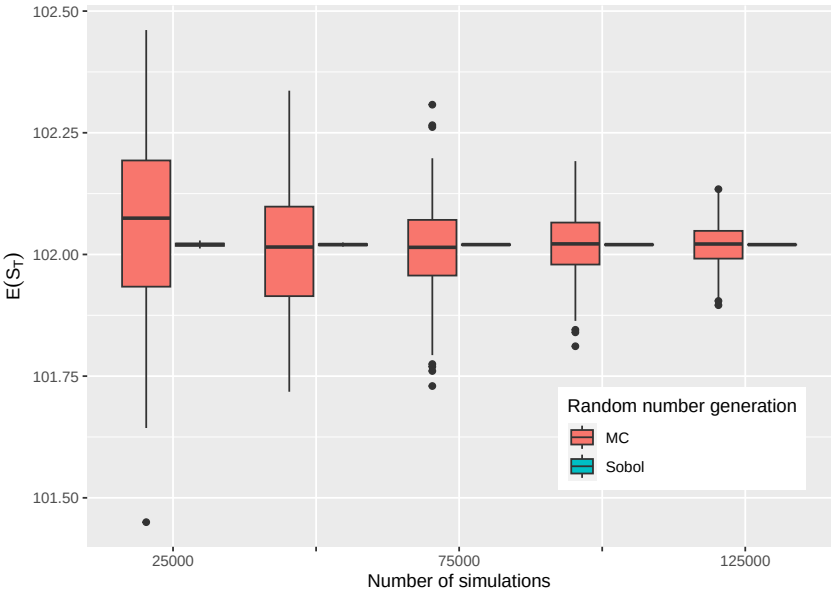


Figure 1.4: Forward as a function of the number of simulations, using simple MC simulations (MC) and Sobol sequences (SOBOL). The mean and s.d. for each number of simulations is computed with 100 trials. The exact value of the Forward is 102.02

Importance Sampling

The principe 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:

$$P = \int_{X \in A} p(X)\phi(X)dX \tag{1.10}$$

$$= \int_{X \in A} p(X)\frac{h(X)}{h(X)}\phi(X)dX \tag{1.11}$$

$$= \int_{X \in A} h(X)\frac{p(X)}{h(X)}\phi(X)dX \tag{1.12}$$

$$= E_h\left[\frac{p(X)}{h(X)}\phi(X)\right] \tag{1.13}$$

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 $S_T^i, i = 1, \ldots, N_A$ 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. We price a European call, 20% out-of-the-money:

```
K <- 120
S.0 <- 100
T <- 0.5
sigma <- .2
r <- .02

price.call <- function(N) {
    Z <- rnorm(N/2, mean=0, sd=1)
    Z <- c(Z, -Z)
    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 <- exp(-r*T)*pmax(S.T-K, 0)[A.index]
    P.call.is <- mean(call.payoff) * nb.A / N
    sd.call.is <- sd(call.payoff) * (nb.A/N) / sqrt(N)

    call.payoff <- exp(-r*T)*pmax(S.T-K, 0)
    P.call <- mean(call.payoff)
    sd.call <- sd(call.payoff) / sqrt(N)
    data.frame(N=rep(N,2), ID=c("MC", "IS"), Mean=c(P.call, P.call.is), SD=c(sd.call, sd.call.is))
  }

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

nb.samples <- seq(from=25000, to=500000, by=25000)
P.call <- foreach(N = nb.samples, .combine="rbind") %dopar% {
  price.call(N)
}
```

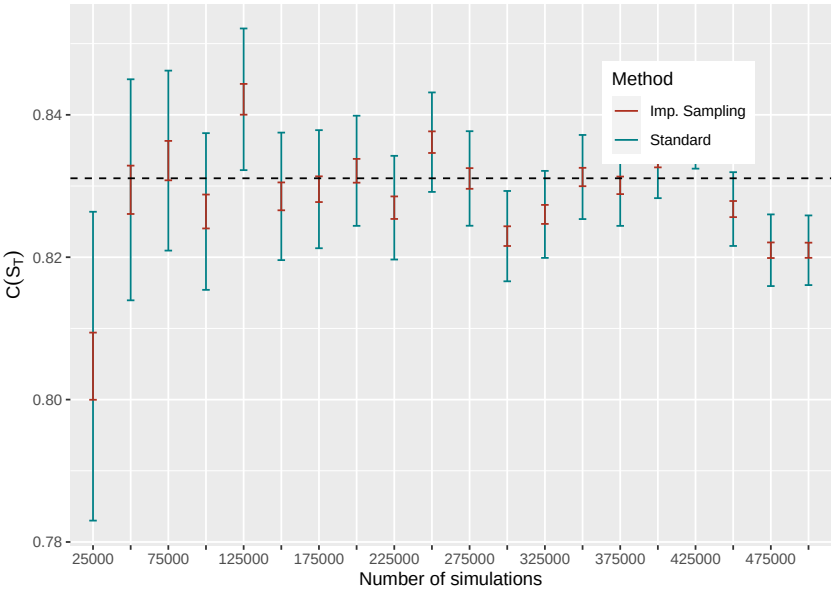


Figure 1.5: Pricing an European Option: standard simulation vs. importance sampling. The analytical value is the dashed line.

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 β being

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

We illustrate the concept by pricing an Asian option (arithmetic average) and using a geometric average option with same observation dates as control variate.

We create 5000 scenarios, with 50 observations per year.

```
dtExpiry <- myDate('01jan2011')
dtStart <- myDate('01jan2010')
nbSteps <- 10
nbPaths <- 300000

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 = F,
    standardization = T, trace = F)
```

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

res <- data.frame(Option="Geom Avg", mean=E.G.asian, sd=se.G.asian)
res <- rbind(res, c("Geom Avg (exact)", G.exact, 0), stringsAsFactors=FALSE)
```

We then price the arithmetic Asian option by simulation.

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

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

res <- rbind(res, c("Arith Avg", E.asian, se.asian), stringsAsFactors=FALSE)
```

Finally, we correct the Monte-Carlo estimate with the control variates, by first estimating the optimal β 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)

res <- rbind(res, c("Arith Avg (CV)", E.asian.cv, se.asian.cv), stringsAsFactors=FALSE)

# 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

res <- rbind(res, c("Arith Avg (TW)", TW, 0), stringsAsFactors=FALSE)
```

Option	mean	sd
Geom Avg	1.27889491500223	0.00883058178610626
Geom Avg (exact)	1.37012196781941	0
Arith Avg	1.5116789301804	0.010080329014342
Arith Avg (CV)	1.61553348029773	0.000743175411475795
Arith Avg (TW)	1.51799410696965	0