Lecture 5: Monte Carlo Method

J. R. Zhang

Department of Computer Science
The University of Hong Kong

Outline

Review of Lecture 4–Implied Volatility

In last lecture, we have learned:

- Implied volatility is a powerful tool and the standard language used in the option world.
- Implied volatilities for different maturities and strikes are usually different.
- Get familiar with the Newton method used to calculate implied volatility:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

.

Review of Lecture 4–Risk Neutrality

- When setting $\mu = r$, we usually say we are in a *risk-neutral* world.
- In risk-neutral world, our stock model becomes:

$$S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}Z}$$

- In this world, the value of a derivative is equal to the expected payoff value discounted for the risk-free interest rate.
- The fair forward level is: $F = \mathbb{E}(S(t)) = S(0)e^{rt}$. Agrees with the non-arbitrage result!
- The European call value is:
  $C(S,t) = e^{-rT}\mathbb{E}(\max(S(T)-K,0)) = SN(d_1) - Ke^{-rT}N(d_2)$.
  Agrees with the non-arbitrage result!
- Why bother with the "risk-neutral" world?

Review of Lecture 4–Risk Neutrality

- The value of an option is the discounted expected payoff at maturity.
- This allows us to develop computational methods for valuating options where analytical formulas are not available.
- Let $f(S(T))$ be the payoff function of some option contract at maturity T. It can be quite complicated and there is no closed-form formulas for the option value function $V(S,0)$.
- We know that, in risk-neutral world, the option value:

$$V(S,0) = e^{-rT}\mathbb{E}(f(S(T))),$$

- In this lecture, we will introduce a new technique to calculate $e^{-rT}\mathbb{E}(f(S(T)))$.

Review of Lecture 4–Monte Carlo Method

Problem

- Consider a general random variable X, whose exact distribution function is unknown.
- But we are interested in the value $\mathbb{E}(X)$.

Monte Carlo Simulation

Let's summarize the basic Monte Carlo simulation method for approximating $\mathbb{E}(X)$:

- We compute M independent samples and calculate the sample mean a_M.
- In order to monitor the approximation error, we also compute the sample variance b_M^2, which allows us to compute the confidence interval.

Variance reduction

- $\frac{b}{\sqrt{M}}$ is often referred to as the *standard error* of the approximation.
- This error is $O(1/\sqrt{M})$, that is, the size of the confidence interval shrinks like the *inverse square root* of the number of samples.
- To reduce the 'error' by a factor of 10 requires a *hundred-fold* increase in the sample size.
- This is a severe limitation of Monte Carlo simulation.
- The size of the confidence interval is directly proportional to the standard deviation of the random variable X under consideration.

Variance reduction

- In practice, it is highly desirable to transform the problem of approximating $\mathbb{E}(X)$ to the problem of approximating $\mathbb{E}(\hat{X})$, which satisfies the following two requirements:

$$\begin{aligned} \mathbb{E}(\hat{X}) &= \mathbb{E}(X), \\ \mathrm{Var}(\hat{X}) &<< \mathrm{Var}(X). \end{aligned}$$

- This idea, known as variance reduction, forms a vital part of practical Monte Carlo algorithms.
- In this course, we introduce two standard approaches to variance reduction:
  - antithetic variate.
  - control variate.

Control Variate

- Let's see how we find $\hat{X}$ in the control variate method.
- The plan is to find some other random variable Y, that is "close" to X with known expected value $E[Y]$.
- Then we define the random variable $\hat{X} \equiv X + \theta(\mathbb{E}(Y) - Y)$ for some $\theta \in \mathbb{R}$.
- Note that $\mathbb{E}(\hat{X}) = \mathbb{E}(X)$, which satisfies our first requirement.

$$\mathbb{E}(\hat{X}) = \mathbb{E}(X) + \theta\mathbb{E}(\mathbb{E}(Y) - Y) = \mathbb{E}(X)$$

- The variance of $\hat{X}$ is

$$\mathrm{var}(\hat{X}) = \mathrm{var}(X - \theta Y) = \mathrm{var}(X) - 2\theta\mathrm{cov}(X, Y) + \theta^2\mathrm{var}(Y).$$

- This is minimized when

$$\theta = \theta_{\text{min}} := \frac{\mathrm{cov}(X, Y)}{\mathrm{var}(Y)}. \tag{1}$$

Control Variate

- Then
$$\mathrm{var}(\hat{X}) = \mathrm{var}(X) - \frac{\mathrm{cov}^2(X, Y)}{\mathrm{var}(Y)} < \mathrm{var}(X).$$
- $\hat{X}$ satisfies our second requirement.
- In practice, we usually don't know $\mathrm{cov}(X, Y)$, but can be estimated from our samples, and use this to minimize the variance of $\hat{X}$.

Control Variate Summary

Summary

The steps for Monte Carlo with control variate:

- Generate the paths of the stock prices.
- Calculate the values of X and Y on all paths: $\bar{X}$ and $\bar{Y}$
- Calculate the sample covariance $\mathrm{cov}(\bar{X}, \bar{Y})$, and calculate θ_{min} where

$$\theta_{\mathsf{min}} = \frac{\mathrm{cov}(\bar{X}, \bar{Y})}{\mathrm{var}(\bar{Y})}$$

- Calculate $\mathbb{E}(\hat{X}) = \mathbb{E}(\bar{X}) + \theta_{\mathsf{min}}(\mathbb{E}(Y) - \mathbb{E}(\bar{Y}))$
- $\mathbb{E}(\hat{X})$ would be our approximation to $\mathbb{E}(X)$

Control Variate in Option Valuation

- Suppose we are employing Monte-Carlo to estimate the value of an option for which a closed-form formula is not readily available.
- Suppose there is a *closely-related* option where there is a closed-form formula for its value.
- The we may take for the control variate this *closely-related* option in the Monte-Carlo simulation. formula.
- The classic example is an arithmetic Asian option, whose payoff at maturity T is

$$\max\left[\frac{1}{n}\sum_{i=1}^{n} S(t_i) - K, 0\right],$$

  where $t_1, t_2, \ldots, t_n$ are a set of pre-defined discrete times.

Control Variate For Arithmetic Asian Option

- Consider a geometric Asian option whose payoff at maturity is

$$\max\left[\left(\prod_{i=1}^{n} S(t_i)\right)^{\frac{1}{n}} - K, 0\right],$$

- For simplicity, let's define a *synthetic* asset $\hat{S}(T) = \left(\prod_{i=1}^{n} S(t_i)\right)^{\frac{1}{n}}$. The payoff at maturity T becomes

$$\max(\hat{S}(T) - K, 0).$$

- In the risk neutral world, the geometric Asian call option value is

$$e^{-rT}\mathbb{E}(\max(\hat{S}(T) - K, 0)).$$

- Note that

$$\prod_{i=1}^{n} S(t_i) = \frac{S(t_n)}{S(t_{n-1})}\left(\frac{S(t_{n-1})}{S(t_{n-2})}\right)^2 \cdots \left(\frac{S(t_1)}{S(t_0)}\right)^n S_0^n$$

Control Variate For Arithmetic Asian Option

- $\frac{S(t_i)}{S(t_{i-1})}$ and $\frac{S(t_j)}{S(t_{j-1})}$ are independent normal random variables, $i \neq j$.
- Then it can be proved that

$$\hat{S}(T) = S_0 e^{(\hat{\mu} - \frac{1}{2}\hat{\sigma}^2)T + \hat{\sigma}\sqrt{T}Z},$$

where $Z \sim N(0,1)$, and

$$\begin{aligned} \hat{\sigma} &= \sigma\sqrt{\frac{(n+1)(2n+1)}{6n^2}} \\ \hat{\mu} &= (r - \frac{1}{2}\sigma^2)\frac{n+1}{2n} + \frac{1}{2}\hat{\sigma}^2. \end{aligned}$$

- Using the derivations from the last lecture, we can deduce the time-zero value of the geometric Asian call option

$$e^{-rT}\left(S_0 e^{\hat{\mu}T} N(\hat{d}_1) - KN(\hat{d}_2)\right), \tag{2}$$

Control Variate For Arithmetic Asian Option

- where $\hat{d}_1 = \hat{d}_2 + \hat{\sigma}\sqrt{T} = \frac{\ln(S_0/K)+(\hat{\mu}+\frac{1}{2}\hat{\sigma}^2)T}{\hat{\sigma}\sqrt{T}}$.
- The time-zero value of the geometric Asian put option can be similarly derived

$$e^{-rT}\left(KN(-\hat{d}_2) - S_0 e^{\hat{\mu}T} N(-\hat{d}_1)\right),$$

- With the above closed-form formulas for geometric Asian call/put options, they can be used as the control variate for the arithmetic Asian call/put options, respectively.

Control Variate For Arithmetic Asian Option

Now let's sketch the control variate method for Arithmetic Asian option:

- Generate M paths of the asset prices at time $t_1, \cdots, t_n$:

$$S^{(j)}(t_i), \text{ where } j = 1, \cdots, M, \ i = 1, \cdots, n$$

.

- Based on the paths of prices, calculate the following quantities:

$$V_{arith} = D \times \frac{1}{M} \sum_{j=1}^{M} \max \left[\frac{1}{n} \sum_{i=1}^{n} S^{(j)}(t_i) - K, 0 \right]$$

$$V_{geo} = D \times \frac{1}{M} \sum_{j=1}^{M} \max \left[\left(\prod_{i=1}^{n} S^{(j)}(t_i) \right)^{\frac{1}{n}} - K, 0 \right]$$

- Calculate the Geometric Asian option value V_{geo}^c with closed-form formula (2)
- Calculate θ using (1)
- Get the Arithmetic Asian Option value $V = V_{arith} + \theta(V_{geo}^c - V_{geo})$.

Control Variate For Arithmetic Asian Option

```
%%%%%% Problem and method parameters %%%%%%%%%
S = 4; E = 4; sigma = 0.25; r = 0.03; T = 1;
Dt = 1e-2; N = T/Dt; M = 1e4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%% Geom Asian exact mean  %%%%%%%%%%%%
sigsqT= sigma^2*T*(N+1)*(2*N+1)/(6*N*N);
muT = 0.5*sigsqT + (r - 0.5*sigma^2)*T*(N+1)/(2*N);

d1 = (log(S/E) + (muT + 0.5*sigsqT))/(sqrt(sigsqT));
d2 = d1 - sqrt(sigsqT);

N1 = normcdf(d1);
N2 = normcdf(d2);

geo =  exp(-r*T)*( S*exp(muT)*N1 - E*N2 );
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Control Variates For Arithmetic Asian Option

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
drift = exp((r-0.5*sigma^2)*Dt);
for i=1:M
 growthFactor = drift * exp(sigma*sqrt(Dt)*randn);
 Spath(1) = S * growthFactor;
 for j = 2:N
    growthFactor = drift * exp(sigma*sqrt(Dt)*randn);
    Spath(j) = Spath(j-1) * growthFactor;
 end

 % Arithmetic mean
 arithMean = mean(Spath);
 arithPayoff(i) = exp(-r*T)*max(arithMean-E,0);  % payoffs

 % Geometric mean
 geoMean = exp((1/N)*sum(log(Spath)));
 geoPayoff(i) = exp(-r*T)*max(geoMean-E,0);      % geo payoffs
end
```

Control Variates For Arithmetic Asian Option

```
% Standard Monte Carlo
Pmean = mean(arithPayoff);
Pstd = std(arithPayoff);
confmc = [Pmean-1.96*Pstd/sqrt(M), Pmean+1.96*Pstd/sqrt(M)]

% Control Variate
covXY = mean(arithPayoff .* geoPayoff)
       - mean(arithPayoff) * mean(geoPayoff);
theta = covXY/var(geoPayoff);

% control variate version
Z = arithPayoff + theta * (geo - geoPayoff);
Zmean = mean(Z);
Zstd = std(Z);
confcv = [Zmean-1.96*Zstd/sqrt(M), Zmean+1.96*Zstd/sqrt(M)]
```

Control Variate From Another Perspective

Control Variate Beyond Monte Carlo

Let's look at control variate method from a broader perspective.

- $\mathbb{E}(\hat{X}) = \mathbb{E}(\bar{X}) + \theta_{\min}(\mathbb{E}(Y) - \mathbb{E}(\bar{Y}))$
- $\mathbb{E}(Y) - \mathbb{E}(\bar{Y})$ is the approximation error for Y
- $\mathbb{E}(\bar{X})$ and $\mathbb{E}(\bar{Y})$ have the same source of approximation error.
- $\mathbb{E}(Y) - \mathbb{E}(\bar{Y})$ can be added to $\mathbb{E}(\bar{X})$ to correct the approximation error.
- Sometimes $\theta_{\min}$ cannot be estimated, in this case, we simply make $\theta_{\min} = 1$ (if X and Y are positively correlated) or $\theta_{\min} = -1$ (if X and Y are negatively correlated).
- The usage of control variate method is not restricted to Monte Carlo simulation.

Antithetic variates

- This technique uses negative correlation. Suppose we are interested in approximating $I = \mathbb{E}(f(U))$, where $U \sim \mathbf{N}(0,1)$, for some function f.
- The standard Monte Carlo estimate is

$$I_M = \frac{1}{M}\sum_{i=1}^{M} f(U_i), \text{ with i.i.d. } U_i \sim \mathbf{N}(0,1).$$

- $(-U) \sim \mathbf{N}(0,1)$, so $I = \mathbb{E}(f(-U))$.
- We now generate another M samples of $f(-U)$ using $-U_i$, $i = 1, 2, \ldots, M$.
- The antithetic alternative is

$$\hat{I}_M = \frac{1}{M}\sum_{i=1}^{M} \frac{f(U_i) + f(-U_i)}{2}, \text{ with i.i.d. } U_i \sim \mathbf{N}(0,1).$$

- $\hat{I}_M$ is a unbiased estimator to $I = \mathbb{E}(f(U))$.

Antithetic variates

- Recall $\mathrm{var}(X+Y) = \mathrm{var}(X) + \mathrm{var}(Y) + 2\mathrm{cov}(X,Y)$.
- We have

$$\begin{aligned}
&\mathrm{var}\left(\frac{f(U_i)+f(-U_i)}{2}\right) \\
&= \frac{\mathrm{var}(f(U_i)) + \mathrm{var}(f(-U_i)) + 2\mathrm{cov}(f(U_i), f(-U_i))}{4} \\
&= \frac{\mathrm{var}(f(U_i)) + \mathrm{cov}(f(U_i), f(-U_i))}{2}
\end{aligned}$$

- An advantage will be gained over simply taking twice as many samples (which would reduce the variance by a factor of two) if

$$\mathrm{cov}(f(U_i), f(-U_i)) < 0.$$

A sufficient condition

- A sufficient condition for $\mathrm{cov}(f(U_i), f(-U_i)) < 0$ is that $f(x)$ is monotonic (either increasing or decreasing).
- Without loss of generality, let's assume that $f(x)$ is monotonically *increasing*, then $f(-x)$ is monotonically *decreasing*. For any two numbers x and y,

$$(f(x) - f(y))(f(-x) - f(-y)) < 0.$$

- Hence if X and Y are i.i.d $\mathbf{N}(0, 1)$, we have

$$\begin{aligned}
0 &> \mathbb{E}((f(X) - f(Y))(f(-X) - f(-Y))) \\
&= \mathbb{E}(f(X)f(-X)) + \mathbb{E}(f(Y)f(-Y)) \\
&\quad - \mathbb{E}(f(X)f(-Y) - f(-X)f(Y)) \\
&= 2\mathbb{E}(f(X)f(-X)) - 2\mathbb{E}(f(X)f(-Y)) \\
&= 2\mathbb{E}(f(X)f(-X)) - 2\mathbb{E}(f(X))\mathbb{E}(f(-Y)) \\
&= 2\mathbb{E}(f(X)f(-X)) - 2\mathbb{E}(f(X))\mathbb{E}(f(-X)) \\
&= 2\mathrm{cov}(f(X)f(-X))
\end{aligned}$$

Antithetic variates in option pricing

Antithetic variates in option pricing

- In this case our generating variables are $Z \sim N(0;1)$, and we construct our antithetic samples by using $-Z$ as well as Z.
- The same considerations apply as in the previous discussion. For a put or a call option some improvement is guaranteed since the payoff functions are monotonic.

Quasi-Monte Carlo

- Both control variate and antithetic methods are product dependent.
- It is difficult to design a generic control variate that works for any product.
- In practice, the quasi-Monte carlo is widely used to improve the convergence.
- Some good books on this topic: Peter Jackel, "Monte Carlo Methods in Finance", and Paul Glasserman, "Monte Carlo Methods in Financial Engineering".

Standard Normal Random Number Generator

Before we discuss the details of the quasi-Monte Carlo method, let's see how to generate samples from standard normal distribution.

- Let $N(x)$ be the cumulative density function of the standard normal random variable $N(0,1)$,

$$N(x) = \int_{-\infty}^{x} \frac{1}{2\pi} e^{-\frac{\xi^2}{2}} d\xi.$$

- The function $N(x)$ is strictly increasing, mapping $\mathbb{R}$ onto $(0,1)$, and so has a strictly increasing inverse function $N^{-1}(y)$ for $y \in (0,1)$.
- In other words, $N(N^{-1}(y)) = y$ for all $y \in (0,1)$.
- Now let Y be a uniformly distributed random variable over $[0,1]$, and set $X = N^{-1}(Y)$. Then X is standard normal random variable.

Standard Normal Random Number Generator

- Whenever $-\infty < a \leq b < +\infty$, we have

$$
\begin{aligned}
P(a < X < b) &= P(a < N^{-1}(Y) < b) \\
&= P(N(a) < N(N^{-1}(Y)) < N(b)) \\
&= P(N(a) < Y < N(b)) \\
&= N(b) - N(a) \\
&= \int_a^b \frac{1}{2\pi} e^{-\frac{\xi^2}{2}} d\xi.
\end{aligned}
$$

- X's density function is exactly the standard normal density function. So X must be a standard normal random variable.
- This theorem is widely used in practice for generating normal random numbers: first from a uniform variable generator we get a sample of the uniform random variable $U(0,1)$, and then apply $N^{-1}(y)$ to get a sample of $N(0,1)$. The inverse function $N^{-1}(y)$ is implemented by some approximation algorithm. One popular one can be found from `http://home.online.no/~pjacklam/notes/invnorm/`.

Standard Normal Random Number Generator

- This theorem is widely used in practice for generating normal random numbers
- First from a uniform variable generator we get a sample of the uniform random variable $U(0,1)$,
- Then apply $N^{-1}(y)$ to get a sample of $N(0,1)$.
- The inverse function $N^{-1}(y)$ is implemented by some approximation algorithm. One popular one can be found from `http://home.online.no/~pjacklam/notes/invnorm/`.
- In the following discussions, we focus on how to generate samples of the uniform random variable $U(0,1)$. This would make our presentation much easier.

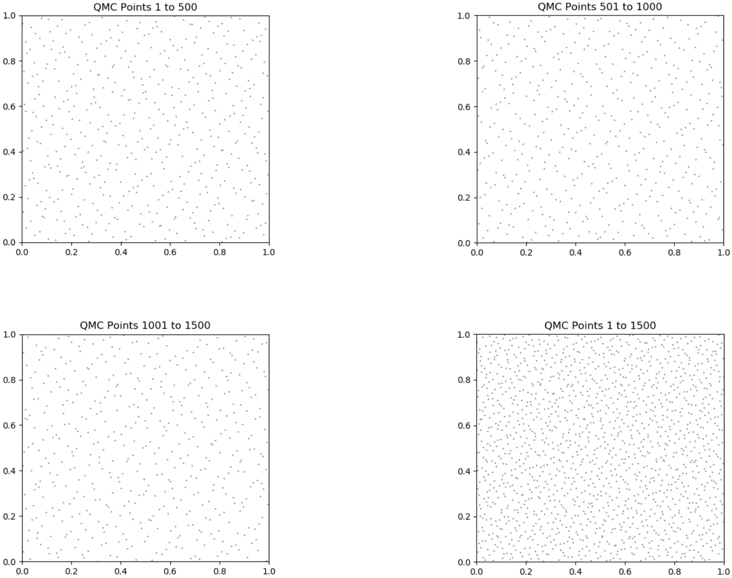
Quasi-Monte Carlo

- Monte Carlo method is based on pseudo random numbers. Quasi-Monte Carlo method uses quasi-random numbers.
- Pseudo-random sequences try to mimic the properties of random sequences.
- Quasi-random numbers, also known as low-discrepancy sequences, are non-random series of numbers. They are much more evenly spread out than random numbers. This makes quasi-Monte Carlo method much more efficient than standard Monte Carlo method.

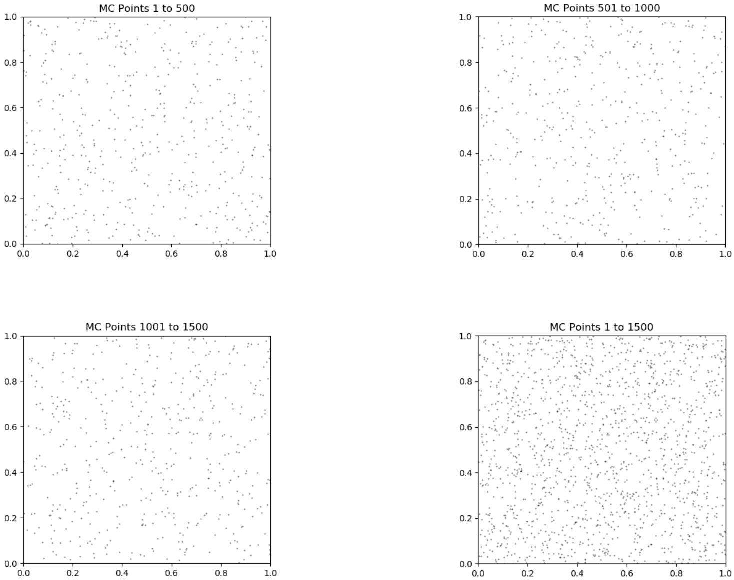
Pseudo-random numbers Vs. Quasi-random numbers

- Pseudo-random sampling scheme tries to mimic random sampling. As it is random sampling, the sampled points could have clusters and gaps.
- A quasi-random sampling scheme tries to fill in gaps between existing samples.
- At each stage of the sampling process, the sampled points are roughly evenly spaced throughout the whole probability space.
- When using the sample mean to estimate the population mean, the standard error is proportional to $\frac{1}{M}$ rather than $\frac{1}{\sqrt{M}}$.
- So quasi-Monte Carlo based on quasi-random sequences is much more efficent.

Quasi-Random Numbers

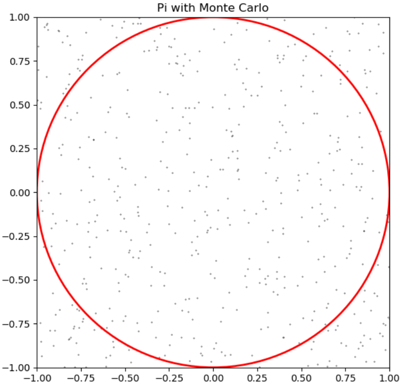


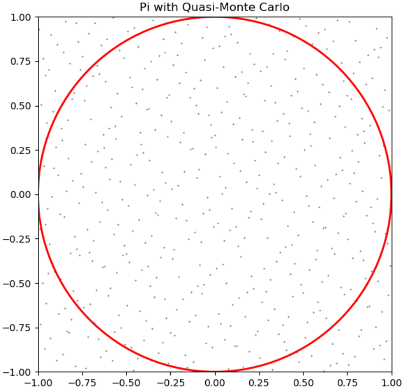
Pseudo Random Numbers



Quasi-Monte Carlo Vs. Standard Monte Carlo

A simple experiment: estimating π with pseudo random numbers/quasi random numbers. Both have 500 sample points. The estimates from both methods are 2.928 and 3.152, respectively.





Quasi-Monte Carlo Vs. Standard Monte Carlo

- Let's now compare QMC and Standard MC using a European call option with the following parameters: $S(0) = 10$; $K = 9$, $r = 0.06$; $\sigma = 0.10$; $T = 1.0$; Its true value is 1.5429374445144521.

Samples	QMC	Standard MC	QMC Error	Standard MC Err
1e2	1.497392	1.603161396	0.04554525	-0.060223951
1e3	1.540451	1.541038273	0.002486175	0.001899171
1e4	1.54245	1.530468142	0.000487776	0.012469302
1e5	1.542925	1.539663281	1.29E-05	0.003274164
1e6	1.542935	1.542342105	2.45E-06	0.000595339

- The starndard MC error gets smallers with greater number of samples in general, but not always.
- The QMC error monotically gets smaller with greater number of samples.
- With the increase of samples, the QMC error goes down much faster than the standard MC errror.

Quasi-Monte Carlo for European Call

```
import numpy as np
import math
from scipy.stats import norm
import ghalton
import black_scholes as bs
#=================================
# option parameters
rate = 0.06;sigma = 0.10;T = 1.0;s0 = 10;K = 9
Ctrue = bs.euro_vanilla_call(s0, K, T, rate, sigma)
#=================================
factor = s0 * np.exp((rate - 0.5 * sigma * sigma) * T)
std = sigma * math.sqrt(T)
#=================================
# generalized halton
# set the random seed
seed = 2000
sequencer = ghalton.GeneralizedHalton(1, seed)
```

Quasi-Monte Carlo for European Call

```
#=================================
aMList = []
aMError = []
for M in [1e2,1e3,1e4,1e5,1e6]:
    M=int(M)
    X = np.array(sequencer.get(M))
    Z = norm.ppf(X)
    factorArray = std * Z
    sArray =  factor * np.exp(factorArray)
    payoffArray = sArray - K
    payoffArray[payoffArray<=0] = 0

    aM = np.mean(payoffArray) * np.exp(-rate * T)
    print('the qmc price is ', aM)
    aMList.append(aM)
    aMError.append(Ctrue - aM)
```

Standard Monte Carlo for European Call

```
import numpy as np
import math
import black_scholes as bs
#=================================
# option parameters
rate = 0.06;sigma = 0.10;T = 1.0;s0 = 10;K = 9
Ctrue = bs.euro_vanilla_call(s0, K, T, rate, sigma)
#=================================
factor = s0 * np.exp((rate - 0.5 * sigma * sigma) * T)
std = sigma * math.sqrt(T)
#=================================
# set the random seed
seed = 1000
np.random.seed(seed)
```

Standard Monte Carlo for European Call

```
#=================================
aMList = []
aMError = []
for M in [1e2,1e3,1e4,1e5,1e6]:
    M=int(M)
    Z = np.random.standard_normal(M)
    factorArray = std * Z
    sArray =  factor * np.exp(factorArray)
    payoffArray = sArray - K
    payoffArray[payoffArray<=0] = 0

    aM = np.mean(payoffArray) * np.exp(-rate * T)
    print('the qmc price is ', aM)
    aMList.append(aM)
    aMError.append(Ctrue - aM)
```

Quasi-Random Number Generators

- There are several different quasi-random number generators: the Halton method, the Nidederreiter method, and the Sobol' method.
- The examples shown in this lecture were generated by a generalized Halton algorithm.
- For problems of low dimensionality, these quasi-random sequences outperform considerably the pseudo-random sequences.
- As the dimensionality increases, this advantage decreases, until around dimensionality = 15, when most of these methods underperform the pseudo-random numbers.
- For suitably initialized Sobol' numbers, it appears much better. That's why Sobol' sequences generator is widely used in industry.
- You can find detailed comparison results from the book "Monte Carlo Methods in Finance" by Peter Jackel.

Dimensionality of A Valuation Problem

- When applying quasi-Monte Carlo method to a valuation problem, have to decide the dimensionality of the problem.
- How to know the dimensionality?
- Dimensionality = number of assets $\times$ number of time steps
- For example:
  - a European option on a single asset: 1
  - an Asian option with n averaging times on a single asset: n
  - A European option on a basket of N assets: N.