Module 2: Monte Carlo Methods

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Overview

In these two lectures, we are concerned with estimating expected discounted payoffs by simulating the solutions of stochastic differential equations

- module 2 – fundamentals for geometric Brownian motion with European payoffs
- module 4 – path-dependent options requiring simulation of entire path

We will not cover the modelling required to come up with the SDE, and in particular will start with the risk-neutral form of the SDE

Geometric Brownian Motion

In the scalar case we have

$$\mathrm{d}S = r\,S\,\mathrm{d}t + \sigma\,S\,\mathrm{d}W$$

and we can use Ito calculus to convert this to

$$\mathrm{d}(\log S) = (r - \tfrac{1}{2}\sigma^2)\,\mathrm{d}t + \sigma\,\mathrm{d}W$$

which can be integrated to give

$$\begin{aligned} \log S(T) &= \log S(0) + (r - \tfrac{1}{2}\sigma^2)\,T + \sigma\,W(T) \\ \Longrightarrow \quad S(T) &= S(0)\ \exp\left((r - \tfrac{1}{2}\sigma^2)\,T + \sigma\,W(T)\right) \end{aligned}$$

Geometric Brownian Motion

In the vector case, each stock has a different volatility σ_i and driving Brownian motion $W_i(t)$, and so

$$S_i(T) = S_i(0)\ \exp\left((r - \tfrac{1}{2}\sigma_i^2)\, T + \sigma_i\, W_i(T)\right)$$

This will be the main application we consider today.

Linkage between stocks comes through correlation in driving Brownian motions

$$\mathbb{E}[\, \mathrm{d}W_i\, \mathrm{d}W_j\,] = \rho_{ij}\ \mathrm{d}t$$

Monte Carlo objectives

What are we trying to achieve with Monte Carlo simulation?

- estimate prices which correspond to expectation of discounted payoff

$$V = \mathbb{E}\left[P(S(T))\right]$$

- estimate price derivatives (Greeks) for hedging,

$$\frac{\partial V}{\partial \theta}$$

  where θ might correspond to initial asset price (delta) or volatility (vega), or some other quantity

Monte Carlo vs. finite differences

Hard to get reliable figures, but my “guesstimate” is that the computational effort (CPU hours) on different methods in the finance industry is split

- 60% Monte Carlo
- 30% finite differences
- 10 % binomial trees and analytic transform methods

So why are Monte Carlo methods used most heavily?

. . . and will it stay that way in the future?

Monte Carlo vs. finite differences

Monte Carlo strengths:

- simple and flexible (with a clear trade-off between simplicity and efficiency)
- easy parallel speedup
- easily able to handle high-dimensional problems (avoids “curse of dimensionality” of finite difference methods)

Monte Carlo weaknesses:

- not as efficient as finite differences for very low dimensions (1-3?)
- not yet efficient for applications with optional exercise (American options, Bermudan options, optimal trading given transaction costs)

Monte Carlo vs. finite differences

What is used in industry?

- FX – finite difference because low-dimensional (1 domestic interest rate, 1 foreign interest rate and 1 exchange rate = 3-dimensional)
- fixed income – MC for LIBOR models because of dimensionality
- energy options – finite difference because low-dimensional and options with conditional exercise
- credit – MC because high-dimensional (multiple companies)
- equities – MC because of high-dimensional baskets

Monte Carlo vs. finite differences

My long-term prediction?

- mathematical modelling likely to become more complex, leading to higher dimensional problems to be solved
- consequently, Monte Carlo methods likely to become more important, rather than less
- improved methods will be developed for American/Bermudan options

Alternative viewpoint?

- sparse grid methods will extend finite difference methods to much higher dimensions
- ability to handle real-world features such as transaction costs will be crucial

Random Number Generation

Monte Carlo simulation starts with random number generation, which often is split into 3 stages:

- generation of independent uniform $(0,1)$ random variables
- conversion into independent Normal $N(0,1)$ random variables
- conversion into correlated Normal $N(0,1)$ random variables

I will focus on what you need to know as a quant
– see *Monte Carlo Methods in Financial Engineering*
by Paul Glasserman for more information

Uniform Random Variables

- Generating "good" uniform random variables is technically complex
- **Never** write your own generator; **always** use a well validated generator from a reputable source
  - Matlab
  - NAG
  - Intel MKL / VSL (Math Kernel / Vector Stats libs)
  - AMD ACML
  - **not** MS Excel, C `rand` function or Numerical Recipes
- What you need to know is what to look for in a good generator

Uniform Random Variables

Pseudo-random number generators use a deterministic (i.e. repeatable) algorithm to generate a sequence of (apparently) random numbers on $(0,1)$ interval.

What defines a good generator?

- a long period – how long it takes before the sequence repeats itself

  2^{32} is not enough – need at least 2^{40}
- various statistical tests to measure “randomness” (Diehard – G. Marsaglia, TestU01 – P. L’Ecuyer)

  well validated software will have gone through these checks

Uniform Random Variables

Practical considerations:

- computational cost – RNG cost can be as large as rest of Monte Carlo simulation
- trivially-parallel Monte Carlo simulation on a compute cluster requires the ability to “skip-ahead” to an arbitrary starting point in the sequence

  first computer gets first 10^6 numbers
  second computer gets second 10^6 numbers, etc

Uniform Random Variables

My favourite: **mrg32k3a**

- developed by Pierre L'Ecuyer
- available in MKL, ACML and NAG libraries
- period $\approx 2^{191} \approx 10^{57}$
- fast skip-ahead makes it well suited to parallel implementation

Mersenne twister is very popular in finance:

- developed by Makoto Matsumoto and Takuji Nishimura
- huge period of $2^{19937}-1$, but I've heard conflicting comments on its statistical properties
- slow skip-ahead makes it tough for parallel implementation

Uniform Random Variables

For more details see

- Intel MKL information

  `www.intel.com/cd/software/products/asmo-na/eng/266864.htm`

- NAG library information

  `www.nag.co.uk/numeric/CL/nagdoc_cl08/pdf/G05/g05_conts.pdf`

- Matlab information

  `www.mathworks.com/moler/random.pdf`

- Wikipedia information

  `en.wikipedia.org/wiki/Random_number_generation`

  `en.wikipedia.org/wiki/List_of_random_number_generators`

  `en.wikipedia.org/wiki/Mersenne_Twister`

Normal Random Variables

To generate $N(0,1)$ Normal random variables, we start with a sequence of uniform random variables on $(0,1)$.

There are then various ways of converting them into $N(0,1)$ Normal variables – I'll mention just two:

- Box-Muller method
- inverse CDF transformation

Normal Random Variables

The Box-Muller method takes y_1, y_2, two independent uniformly distributed random variables on $(0,1)$ and defines

$$\begin{aligned} x_1 &= \sqrt{-2\log(y_1)}\ \cos(2\pi y_2) \\ x_2 &= \sqrt{-2\log(y_1)}\ \sin(2\pi y_2) \end{aligned}$$

It can be proved that x_1 and x_2 are $N(0,1)$ random variables, and independent.

A $\log$, $\cos$ and $\sin$ operation per 2 Normals makes this a slightly expensive method.

Normal Random Variables

The transformation method takes y, uniformly distributed on $(0,1)$, and defines

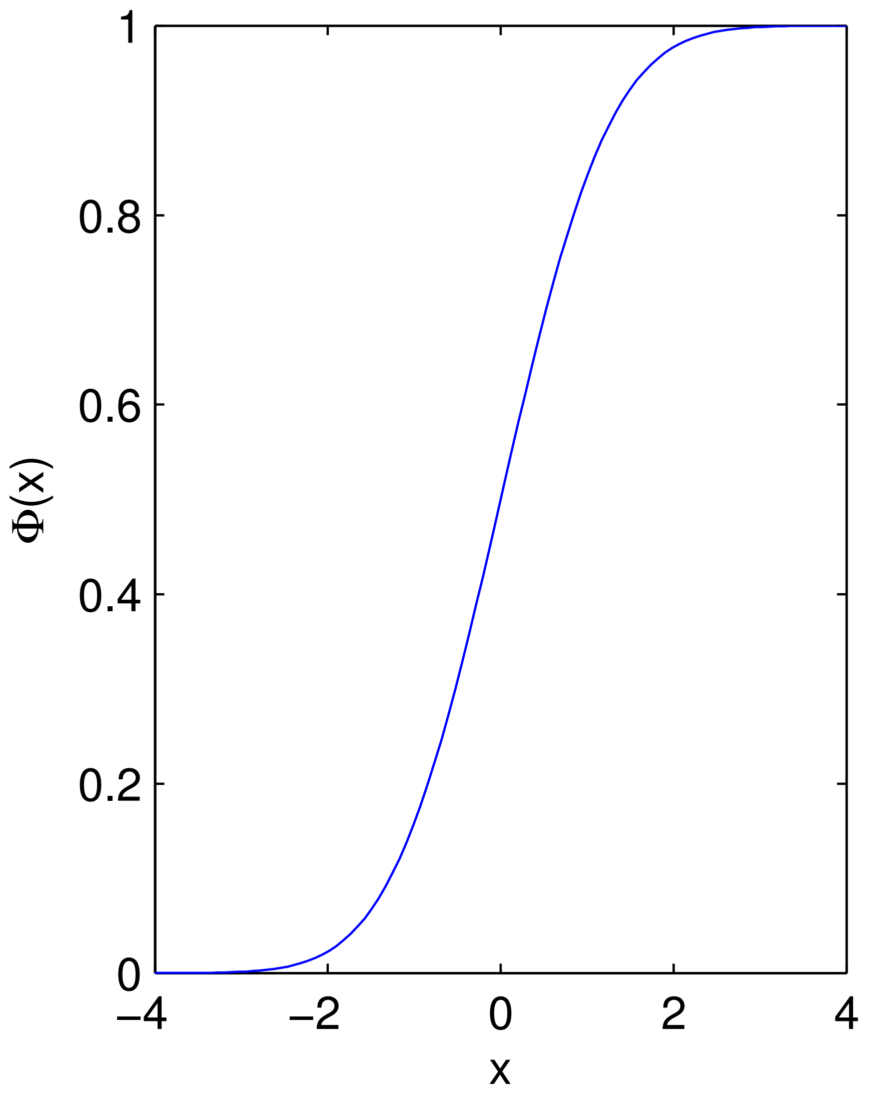
$$x = \Phi^{-1}(y),$$

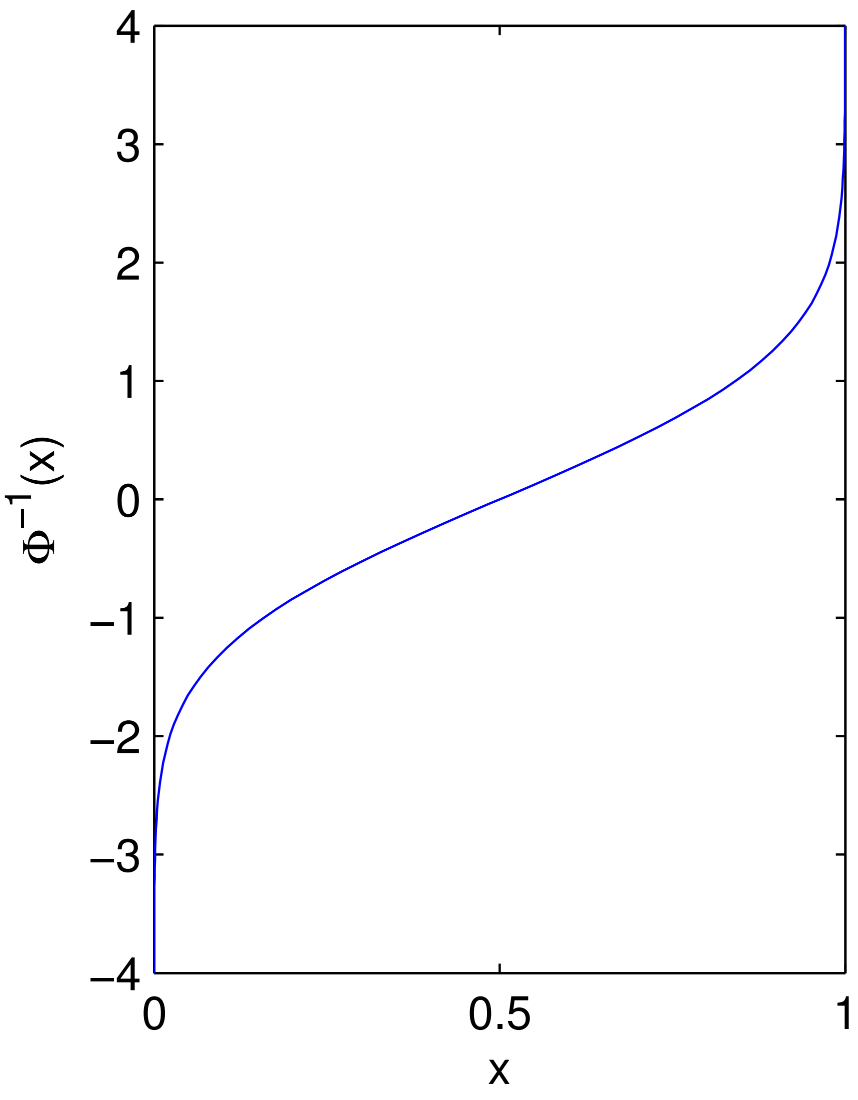
where $\Phi(x)$ is the Normal cumulative distribution function.

$\Phi^{-1}(y)$ is approximated in software in a very similar way to the implementation of $\cos, \sin, \log$, so this is just as accurate as the other methods.

It is also a more flexible approach because we'll need $\Phi^{-1}(y)$ later for stratified sampling and quasi-Monte Carlo methods.

Normal Random Variables





Normal Random Variables

Some useful weblinks:

- `home.online.no/~pjacklam/notes/invnorm/`
  code for Φ^{-1} function in many different languages
- `lib.stat.cmu.edu/apstat/241/`
  single and double precision code in FORTRAN (coming soon in next version of NAG libraries)
- `en.wikipedia.org/wiki/Normal_distribution`
  Wikipedia definition of Φ matches mine
- `mathworld.wolfram.com/NormalDistribution.html`
  `mathworld.wolfram.com/DistributionFunction.html`
  Good Mathworld items, but their definition of Φ is slightly different; they call the cumulative distribution function $D(x)$.

Normal Random Variables

The Normal CDF $\Phi(x)$ is related to the error function $\mathrm{erf}(x)$ through

$$\Phi(x) = \tfrac{1}{2} + \tfrac{1}{2}\mathrm{erf}(x/\sqrt{2}) \quad\Longrightarrow\quad \Phi^{-1}(y) = \sqrt{2}\,\mathrm{erf}^{-1}(2y-1)$$

This is the function I use in Matlab code when `norminv` is not available:

```
% x = ncfinv(y)
%
% inverse Normal CDF

function x = ncfinv(y)

x = sqrt(2)*erfinv(2*y-1);
```

Correlated Normal Random Variables

The final step is to generate a vector of Normally distributed variables with a prescribed covariance matrix.

Suppose x is a vector of independent $N(0,1)$ variables, and define a new vector $y = L\,x$.

Each element of y is Normally distributed, $\mathbb{E}[y] = L\,\mathbb{E}[x] = 0$, and

$$\mathbb{E}[y\,y^T] = \mathbb{E}[L\,x\,x^T\,L^T] = L\,\mathbb{E}[x\,x^T]\,L^T = L\,L^T.$$

since $\mathbb{E}[x\,x^T] = I$ because

- elements of x are independent $\Longrightarrow \mathbb{E}[x_i\,x_j] = 0$ for $i \neq j$
- elements of x have unit variance $\Longrightarrow \mathbb{E}[x_i^2] = 1$

Correlated Normal Random Variables

To get $\mathbb{E}[y\, y^T] = \Sigma$, we need to find L such that

$$L\, L^T = \Sigma$$

L is not uniquely defined, but any choice will give correct correlated distribution.

Simplest choice is to use a Cholesky factorization in which L is lower-triangular, with a positive diagonal. In MATLAB, use the **chol** function.

In Module 6, will use other factorisations with quasi-Monte Carlo methods.

Final RNG advice

- **always** use mathematical libraries as much as possible
- usually they will give you uncorrelated Normals, and you have to convert these into correlated Normals
- later with stratified sampling and quasi-Monte Carlo methods, we will use the inverse cumulative Normal distribution to convert (quasi-)uniforms into (quasi-)Normals

Expectation and Integration

If x is a random variable uniformly distributed on $[0,1]$ then the expectation of a function $f(x)$ is equal to its integral:

$$\overline{f} = \mathbb{E}[f(x)] = I[f] = \int_0^1 f(x)\,\mathrm{d}x.$$

The generalisation to a d-dimensional “cube” $I^d = [0,1]^d$, is

$$\overline{f} = \mathbb{E}[f(x)] = I[f] = \int_{I^d} f(x)\,\mathrm{d}x.$$

Thus the problem of finding expectations in finance is directly connected to the problem of numerical quadrature (integration), often in very large dimensions.

Expectation and Integration

Suppose we have a sequence x_n of independent samples from the uniform distribution.

An approximation to the expectation/integral is given by

$$I_N[f] = N^{-1} \sum_{n=1}^{N} f(x_n).$$

Two key features:

- Unbiased: $\mathbb{E}\left[I_N[f]\right] = I[f]$
- Convergent: $\lim_{N\to\infty} I_N[f] = I[f]$

Expectation and Integration

In general, define

- error $\varepsilon_N(f) = I[f] - I_N[f]$
- bias $= \mathbb{E}[\varepsilon_N(f)]$
- RMSE, “root-mean-square-error” $= \sqrt{\mathbb{E}[(\varepsilon_N(f))^2]}$

The Central Limit Theorem proves that for large N

$$\varepsilon_N(f) \sim \sigma\, N^{-1/2}\, Z$$

with Z a $N(0,1)$ random variable and σ^2 the variance of f:

$$\sigma^2 = \mathbb{E}[(f-\overline{f})^2] = \int_{I^d} \left(f(x) - \overline{f}\right)^2\, \mathrm{d}x.$$

Expectation and Integration

More precisely, provided σ is finite, then as $N \longrightarrow \infty$,

$$\text{CDF}(N^{1/2}\sigma^{-1}\varepsilon_N) \longrightarrow \text{CDF}(Z)$$

so that

$$\mathbb{P}\left[N^{1/2}\sigma^{-1}\varepsilon_N < s\right] \;\longrightarrow\; \mathbb{P}\left[Z < s\right] \;=\; \Phi(s)$$

and

$$\mathbb{P}\left[\left|N^{1/2}\sigma^{-1}\varepsilon_N\right| > s\right] \;\longrightarrow\; \mathbb{P}\left[|Z| > s\right] \;=\; 2\,\Phi(-s)$$

$$\mathbb{P}\left[\left|N^{1/2}\sigma^{-1}\varepsilon_N\right| < s\right] \;\longrightarrow\; \mathbb{P}\left[|Z| < s\right] \;=\; 1 - 2\,\Phi(-s)$$

Expectation and Integration

Given N samples, the empirical variance is

$$\widetilde{\sigma}^2 = N^{-1} \sum_{n=1}^{N} \left(f(x_n) - I_N\right)^2 = I_N^{(2)} - (I_N)^2$$

where

$$I_N = N^{-1} \sum_{n=1}^{N} f(x_n), \qquad I_N^{(2)} = N^{-1} \sum_{n=1}^{N} \left(f(x_n)\right)^2$$

$\widetilde{\sigma}^2$ is a slightly biased estimator for σ^2; an unbiased estimator is

$$\widehat{\sigma}^2 = (N-1)^{-1} \sum_{n=1}^{N} \left(f(x_n) - I_N\right)^2 = \frac{N}{N-1} \left(I_N^{(2)} - (I_N)^2\right)$$

Expectation and Integration

Objective: want an accuracy of $\bar{\varepsilon}$ with confidence c.
i.e. $|\varepsilon| < \bar{\varepsilon}$ with probability c.

How many samples do we need to use?

Recall,

$$\mathbb{P}\left[N^{1/2}\sigma^{-1}|\varepsilon| < s \right] \approx 1 - 2\,\Phi(-s),$$

so define function $s(c)$ such that

$$1 - 2\,\Phi(-s) = c \quad \Longleftrightarrow \quad s = -\Phi^{-1}((1-c)/2)$$

Expectation and Integration

c	0.683	0.9545	0.9973	0.99994
s	1.0	2.0	3.0	4.0

Then $|\varepsilon| < N^{-1/2}\,\sigma\, s(c)$ with probability c, so to get $|\varepsilon| < \overline{\varepsilon}$ we can put

$$N^{-1/2}\,\widehat{\sigma}\, s(c) = \overline{\varepsilon} \quad \Longrightarrow \quad N = \left(\frac{\widehat{\sigma}\, s(c)}{\overline{\varepsilon}}\right)^2 .$$

Note: twice as much accuracy requires 4 times as many samples.

Expectation and Integration

How does Monte Carlo integration compare to grid based methods for d-dimensional integration?

MC error is proportional to $N^{-1/2}$ independent of the dimension.

If the integrand is sufficiently smooth, trapezoidal integration with $M = N^{1/d}$ points in each direction has

$$\text{Error} \;\propto\; M^{-2} \;=\; N^{-2/d}$$

This scales better than MC for $d < 4$, but worse for $d > 4$.
i.e. MC is better at handling high dimensional problems.

Applications

Geometric Brownian motion for single asset:

$$S(T) = S_0 \, \exp\left((r - \tfrac{1}{2}\sigma^2)T + \sigma\, W(T) \right)$$

$W(T)$ has a Normal distribution with mean 0, variance T; from this we will calculate the risk-neutral expectation for

$$V = \mathbb{E}\left[f(S(T)) \right]$$

Applications

We can put

$$W(T) = \sqrt{T}\, Y = \sqrt{T}\, \Phi^{-1}(U)$$

where Y is a $N(0,1)$ random variable, and U is uniformly distributed on $[0,1]$.

Thus

$$V = \mathbb{E}\left[f(S(T))\right] = \int_0^1 f(S(T))\, \mathrm{d}U,$$

with

$$\begin{aligned} S(T) &= S_0\, \exp\left((r - \tfrac{1}{2}\sigma^2)T + \sigma\sqrt{T}\, Y\right) \\ &= S_0\, \exp\left((r - \tfrac{1}{2}\sigma^2)T + \sigma\sqrt{T}\, \Phi^{-1}(U)\right) \end{aligned}$$

Applications

For the European call option,

$$f(S) = \exp(-rT)\ (S-K)^+$$

while for the European put option

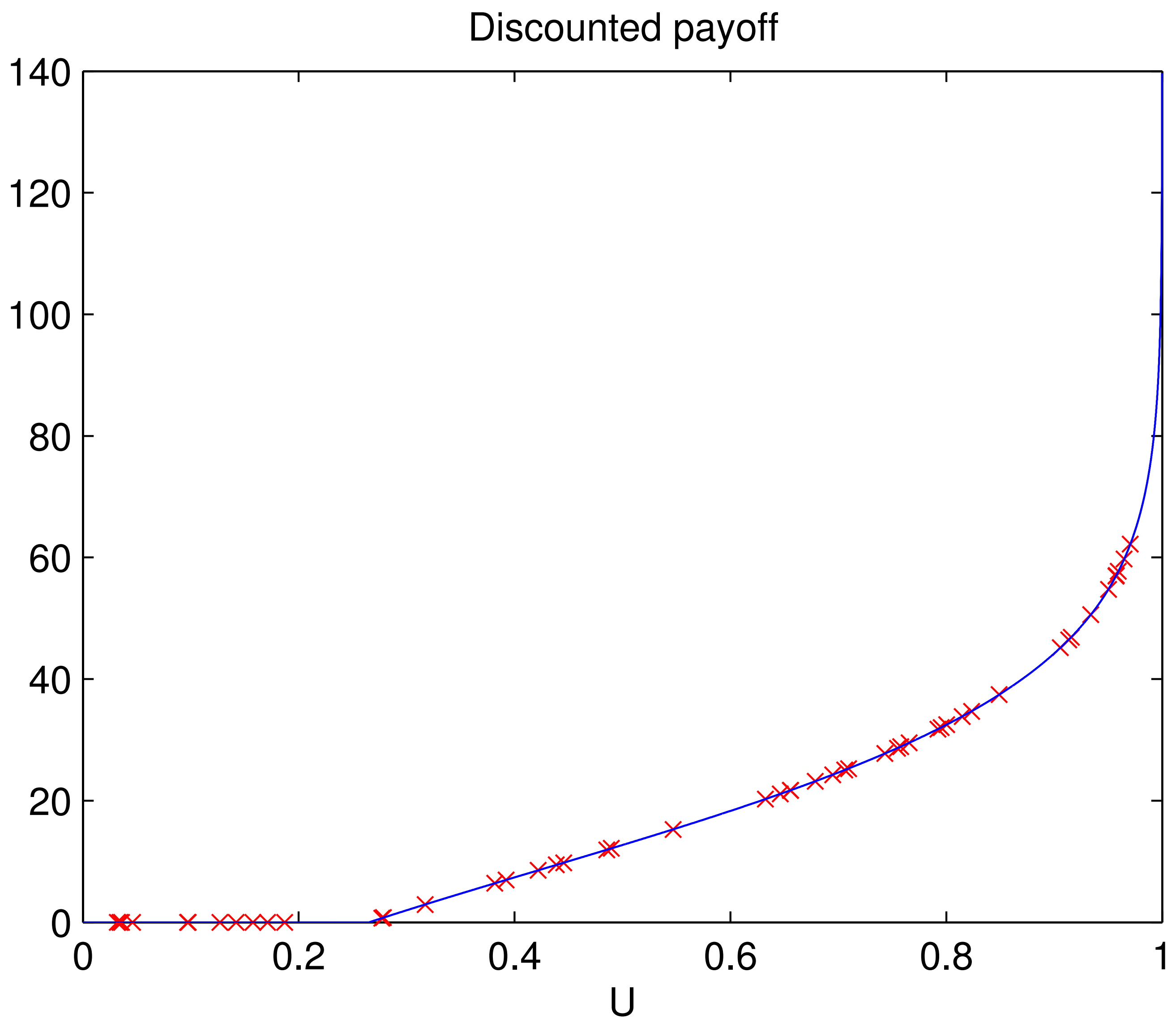
$$f(S) = \exp(-rT)\ (K-S)^+$$

where K is the strike price.

For numerical experiments we will consider a European call with $\quad r=0.05, \quad \sigma = 0.2, \quad T=1, \quad S_0=110, \quad K=100.$

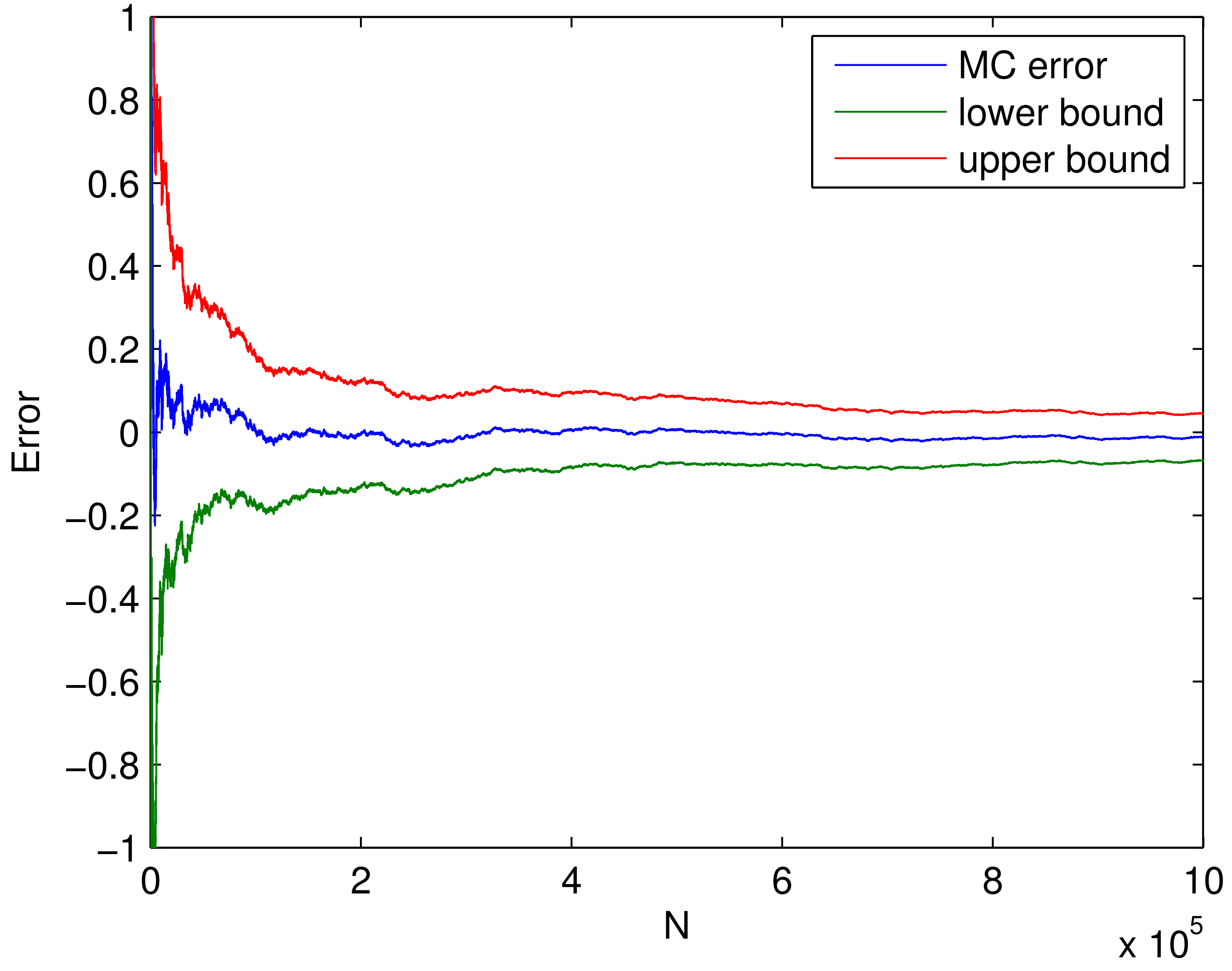
The analytic value is known for comparison.

Applications



Applications

MC calculation with up to 10^6 paths; true value = 17.663



Applications

The upper and lower bounds are given by

$$\text{Mean} \pm \frac{3\,\widetilde{\sigma}}{\sqrt{N}},$$

so more than a 99.7% probability that the true value lies within these bounds.

Applications

MATLAB code:

```
r=0.05;  sig=0.2;  T=1;  S0=110;  K=100;
N = 1:1000000;
U = rand(1,max(N));     % uniform random variable
Y = norminv(U);         % inverts Normal cum. fn.
S = S0*exp((r-sig^2/2)*T + sig*sqrt(T)*Y);
F = exp(-r*T)*max(0,S-K);

sum1 = cumsum(F);       % cumulative summation of
sum2 = cumsum(F.^2);    % payoff and its square
val  = sum1./N;
rms  = sqrt(sum2./N - val.^2);
```

Applications

```
err = european_call(r,sig,T,S0,K,'value') - val;

plot(N,err,                        ...
     N,err-3*rms./sqrt(N),         ...
     N,err+3*rms./sqrt(N))
axis([0 length(N) -1 1])
xlabel('N'); ylabel('Error')
legend('MC error','lower bound','upper bound')
```

Applications

New application: basket option

European call for arithmetic average of M stocks which are correlated so that

$$\mathrm{d}S_i = r\, S_i\, \mathrm{d}t + \sigma_i S_i \mathrm{d}W_i$$

with the different $\mathrm{d}W_i$ <u>not</u> independent.

As before, get

$$S_i(T) = S_i(0)\ \exp\left((r - \tfrac{1}{2}\sigma_i^2) T + \sigma_i W_i(T) \right)$$

Applications

If $\sigma_i W_i(T)$ have covariance matrix Σ, then use Cholesky factorisation $LL^T = \Sigma$ to get

$$S_i(T) = S_i(0)\ \exp\left((r - \tfrac{1}{2}\sigma_i^2)T + \sum_j L_{ij} Y_j \right)$$

where Y_j are independent $N(0,1)$ random variables.

Each Y_i can in turn be expressed as $\Phi^{-1}(U_i)$ where the U_i are uniformly, and independently, distributed on $[0,1]$.

Applications

The payoff is

$$f = \exp(-rT) \left(\frac{1}{M} \sum_i S_i - K \right)^+$$

and so the expectation can be written as the M-dimensional integral

$$\int_{I^M} f(U) \, \mathrm{d}U.$$

This is a good example for Monte Carlo simulation – cost scales linearly with the number of stocks, whereas it would be exponential for grid-based numerical integration.

Final Words on Basics

- Monte Carlo quadrature is straightforward and robust
- Confidence bounds can be obtained as part of the calculation
- Can calculate the number of samples N needed for chosen accuracy
- Much more efficient than grid-based methods for high dimensions
- Accuracy = $O(N^{-1/2})$, CPU time = $O(N)$

$\Longrightarrow$ accuracy = $O(\text{CPU time}^{-1/2})$

$\Longrightarrow$ CPU time = $O(\text{accuracy}^{-2})$

Variance Reduction

Monte Carlo starts as a very simple method; much of the complexity in practice comes from trying to reduce the variance, to reduce the number of samples that have to be simulated to achieve a given accuracy.

- antithetic variables
- control variates
- importance sampling
- stratified sampling (see Glasserman)
- Latin hypercube (see Glasserman)
- quasi-Monte Carlo (module 6)

Review of elementary results

If a, b are random variables, and λ, μ are constants, then

$$
\begin{aligned}
\mathbb{E}[a+\mu] &= \mathbb{E}[a]+\mu \\
\mathbb{V}[a+\mu] &= \mathbb{V}[a] \\
\mathbb{E}[\lambda\, a] &= \lambda\, \mathbb{E}[a] \\
\mathbb{V}[\lambda\, a] &= \lambda^2\, \mathbb{V}[a] \\
\mathbb{E}[a+b] &= \mathbb{E}[a]+\mathbb{E}[b] \\
\mathbb{V}[a+b] &= \mathbb{V}[a]+2\,\text{Cov}[a,b]+\mathbb{V}[b]
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbb{V}[a] &\equiv \mathbb{E}\left[(a-\mathbb{E}[a])^2\right] = \mathbb{E}\left[a^2\right]-(\mathbb{E}[a])^2 \\
\text{Cov}[a,b] &\equiv \mathbb{E}\Big[(a-\mathbb{E}[a])\,(b-\mathbb{E}[b])\Big]
\end{aligned}
$$

Antithetic variables

The simple estimator from the last lecture has the form

$$N^{-1} \sum_i f(W^{(i)})$$

where $W^{(i)}$ is the value of the random Weiner variable $W(T)$ at maturity.

$W(T)$ has a symmetric probability distribution so $-W(T)$ is just as likely.

Antithetic variables

Antithetic estimator replaces $f(W^{(i)})$ by

$$\overline{f}^{(i)} = \tfrac{1}{2}\left(f(W^{(i)}) + f(-W^{(i)})\right)$$

Clearly still unbiased since

$$\mathbb{E}[\overline{f}] = \tfrac{1}{2}\left(\mathbb{E}[f(W)] + \mathbb{E}[f(-W)]\right) = \mathbb{E}[f(W)]$$

The variance is given by

$$\begin{aligned}\mathbb{V}[\overline{f}] &= \tfrac{1}{4}\left(\mathbb{V}[f(W)] + 2\,\mathsf{Cov}[f(W), f(-W)] + \mathbb{V}[f(-W)]\right)\\ &= \tfrac{1}{2}\left(\mathbb{V}[f(W)] + \mathsf{Cov}[f(W), f(-W)]\right)\end{aligned}$$

Antithetic variables

The variance is always reduced, but the cost is almost doubled, so net benefit only if Cov$[f(W), f(-W)] < 0$.

Two extremes:

- A linear payoff, $f = a + b\,W$, is integrated exactly since $\overline{f} = a$ and Cov$[f(W), f(-W)] = -\mathbb{V}[f]$
- A symmetric payoff $f(W) = f(-W)$ is the worst case since Cov$[f(W), f(-W)] = \mathbb{V}[f]$

General assessment – usually not very helpful, but can be good in particular cases where the payoff is nearly linear

Control Variates

Suppose we want to approximate $\mathbb{E}[f]$ using a simple Monte Carlo average $\overline{f}$.

If there is another payoff g for which we know $\mathbb{E}[g]$, can use $\overline{g} - \mathbb{E}[g]$ to reduce error in $\overline{f} - \mathbb{E}[f]$.

How? By defining a new estimator

$$\widehat{f} = \overline{f} - \lambda\,(\overline{g} - \mathbb{E}[g])$$

Again unbiased since $\mathbb{E}[\widehat{f}] = \mathbb{E}[\overline{f}] = \mathbb{E}[f]$

Control Variates

For a single sample,

$$\mathbb{V}[f - \lambda\,(g - \mathbb{E}[g])] = \mathbb{V}[f] - 2\,\lambda\,\text{Cov}[f,g] + \lambda^2\,\mathbb{V}[g]$$

For an average of N samples,

$$\mathbb{V}[\overline{f} - \lambda\,(\overline{g} - \mathbb{E}[g])] = N^{-1}\left(\mathbb{V}[f] - 2\,\lambda\,\text{Cov}[f,g] + \lambda^2\,\mathbb{V}[g]\right)$$

To minimise this, the optimum value for λ is

$$\lambda = \frac{\text{Cov}[f,g]}{\mathbb{V}[g]}$$

Control Variates

The resulting variance is

$$N^{-1}\,\mathbb{V}[f]\left(1-\frac{(\mathrm{Cov}[f,g])^2}{\mathbb{V}[f]\,\mathbb{V}[g]}\right) = N^{-1}\,\mathbb{V}[f]\left(1-\rho^2\right)$$

where ρ is the correlation between f and g.

The challenge is to choose a good g which is well correlated with f – the covariance, and hence the optimal λ, can be estimated from the data.

Control Variates

Possible choices:

- for European call option (ignoring its known value) could use $g = S$ since

$$\mathbb{E}[S(T)] = \exp(rT)\; S(0)$$

- for a general European payoff $f(S)$ could use a combination of put and call options

The idea can also be taken further using multiple control variates.

General assessment – can be very effective, depending on the application

Importance Sampling

Importance sampling involves a change of probability measure. Instead of taking X from a distribution with p.d.f. $p_1(X)$, we instead take it from a different distribution with p.d.f. $p_2(X)$.

$$\begin{aligned} \mathbb{E}_1[f(X)] &= \int f(X)\, p_1(X)\, \mathrm{d}X \\ &= \int f(X)\, \frac{p_1(X)}{p_2(X)}\, p_2(X)\, \mathrm{d}X \\ &= \mathbb{E}_2[f(X)\; R(X)] \end{aligned}$$

where $R(X) = p_1(X)/p_2(X)$ is the Radon-Nikodym derivative.

Importance Sampling

We want the new variance $\mathbb{V}_2[f(X)\ R(X)]$ to be smaller than the old variance $\mathbb{V}_1[f(X)]$.

How do we achieve this? Ideal is to make $f(X)R(X)$ constant, so its variance is zero.

More practically, make $R(X)$ small where $f(X)$ is large, and make $R(X)$ large where $f(X)$ is small.

Small $R(X) \iff$ large $p_2(X)$ relative to $p_1(X)$, so more random samples in region where $f(X)$ is large.

Particularly important for rare event simulation where $f(X)$ is zero almost everywhere.

Importance Sampling

Really simple example of the problem with rare events: suppose random variable X takes value 1 with probability $\delta \ll 1$ and is otherwise 0.

$$\mathbb{E}[X] = \delta$$

$$\mathbb{V}[X] = \mathbb{E}[X^2] - \left(\mathbb{E}[X]\right)^2 = \delta - \delta^2$$

Hence,

$$\frac{\sqrt{\mathbb{V}[X]}}{\mathbb{E}[X]} = \sqrt{\frac{1-\delta}{\delta}} \approx \sqrt{\frac{1}{\delta}}$$

If we want the relative error to be less than ε, the number of samples required is $O(\varepsilon^{-2}\delta^{-1})$.

Importance Sampling

Digital put option:

$$P = \exp(-rT)\ H(K - S(T)) = \exp(-rT)\ H(\log K - \log S(T))$$

where

$$X = \log S(T) = \log S(0) + (r - \tfrac{1}{2}\sigma^2)\, T + \sigma\, W(T)$$

is Normally distributed with p.d.f.

$$\phi_1(X) = \frac{1}{\sqrt{2\pi\sigma^2 T}}\ \exp\left(\frac{-(x-\mu)^2}{2\sigma^2 T} \right)$$

with $\mu = \log S(0) + (r - \frac{1}{2}\sigma^2)\, T$.

Importance Sampling

A digital put option with very low strike (e.g. $K = 0.4\, S(0)$) is sometimes used as a hedge for credit derivatives.

If the stock price falls that much, there is a strong possibility of credit default.

Problem: this is a rare event. The probability that $S(T) < K$ can be very low, maybe less than 1%, leading to a very high r.m.s. error relative to the true price.

Solution: importance sampling, adjusting either mean or volatility

Importance Sampling

Approach 1: change the mean from μ_1 to $\mu_2 < \mu_1$ by using

$$X = \mu_2 + \sigma\, W(T)$$

The Radon-Nikodym derivative is

$$\begin{aligned} R(X) &= \exp\left(\frac{-(X-\mu_1)^2}{2\sigma^2 T}\right) / \exp\left(\frac{-(X-\mu_2)^2}{2\sigma^2 T}\right) \\ &= \exp\left(\frac{(X-\frac{1}{2}(\mu_1+\mu_2))(\mu_1-\mu_2)}{\sigma^2 T}\right) \\ &> 1 \text{ for } X > \tfrac{1}{2}(\mu_1+\mu_2) \\ &< 1 \text{ for } X < \tfrac{1}{2}(\mu_1+\mu_2) \end{aligned}$$

Choosing $\mu_2 = \log K$ means half of samples are below $\log K$ with very small $R(X) \Longrightarrow$ large variance reduction

Importance Sampling

Approach 2: change the volatility from σ_1 to $\sigma_2 > \sigma_1$ by using

$$X = \mu + \sigma_2\, W(T)$$

The Radon-Nikodym derivative is

$$\begin{aligned}
R(X) &= \sigma_1^{-1} \exp\left(\frac{-(X-\mu)^2}{2\sigma_1^2 T} \right) \Big/ \sigma_2^{-1} \exp\left(\frac{-(X-\mu)^2}{2\sigma_2^2 T} \right) \\
&= \frac{\sigma_2}{\sigma_1} \exp\left(\frac{-(X-\mu)^2(\sigma_2^2-\sigma_1^2)}{2\sigma_1^2\sigma_2^2 T} \right) \\
&> 1 \text{ for small } X \\
&\ll 1 \text{ for large } X
\end{aligned}$$

This is good for applications where both tails are important
– not as good in this application.

Final Words on Variance Reduction

- antithetic variables – generic and easy to implement but limited effectiveness
- control variates – easy to implement and can be very effective but requires careful choice of control variate in each case
- importance sampling – very useful for applications with rare events, but needs to be fine-tuned for each application

Overall, a tradeoff between simplicity and generality on one hand, and efficiency and programming effort on the other.