Stochastic Simulation

End-of-course practical MT 2022

This practical is to implement some of the methods covered in the course. You can do this in your preferred language (python, Matlab, C/C++, R). Some sample code is available, but not always in all languages.

Please work on this as a group of 3 or 4.

To submit: A document containing plots of the main results with a discussion of the findings in relation to the theory; the code used to produce the plots.

1. Let U be uniformly distributed on $[0, 1]$. You are to use Monte Carlo simulation to estimate the value of

$$\overline{f} = \mathbb{E}[f(U)] = \int_0^1 f(U)\ \mathrm{d}U$$

where

$$f(x) = x\ \cos \pi x.$$

(a) Calculate analytically the exact value for $\overline{f}$ and

$$\sigma^2 = \mathbb{E}[(f(U) - \overline{f})^2] = \int_0^1 (f(U) - \overline{f})^2\ \mathrm{d}U.$$

(b) Write a program to compute

$$Y_m = N^{-1} \sum_{n=1}^{N} f(U^{(m,n)})$$

for 1000 different sets of 1000 independent random variables $U^{(m,n)}$.

(c) Sort the Y_m into ascending order, and then plot $C_m = (m - 1/2)/1000$ versus Y_m – this is the numerical cumulative distribution function.

Superimpose on the same plot the cumulative distribution function you would expect from the Central Limit Theorem, and comment on your results.

You may like to experiment by trying larger or smaller sets of points to improve your understanding of the asymptotic behaviour described by the CLT.

(For those doing experiments in C++, I suggest you do the plotting in python.)

(d) Modify your code to use a single set of 10^6 random numbers, and plot

$$Y_N = N^{-1} \sum_{n=1}^{N} f(U^{(n)})$$

versus N for $N = 10^3 - 10^6$. This should demonstrate the convergence to the true value predicted by the Strong Law of Large Numbers.

For each N, also compute an unbiased estimate for the variance σ^2 and hence add to the plot upper and lower confidence bounds based on 3 standard deviations of the variation in the mean.

Add a line corresponding to the true value. Does this lie inside the bounds?

2. Now consider a European call option in which the final value of the underlying is

$$S(T) = S(0)\ \exp\left((r - \tfrac{1}{2}\sigma^2)T + \sigma W(T)\right)$$

where

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

with X being a unit Normal, or U a uniform $(0, 1)$ random variable.

The payoff function is

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

and the constants are $r = 0.05, \sigma = 0.2, S(0) = 100, K = 100$.

The analytic value is given by the routine `european_call` available from the course webpage; read its header to see how to call it.

(There is no need to compute the analytic variance as in part a) in the previous question; just use the unbiased estimator.)

Investigate the following forms of variance reduction:

(a) First, try antithetic variables using $\frac{1}{2}\left(f(W) + f(-W)\right)$ where W is the value of the underlying Brownian motion at maturity.

What is the estimated correlation between $f(W)$ and $f(-W)$? How much variance reduction does this give?

(b) Second, try using $\exp(-rT)\, S(T)$ as a control variate, noting that its expected value is $S(0)$.

Again, how much variance reduction does this give?

(c) For the case of a digital put option,

$$P = \exp(-r\,T)\ H(K - S(T))$$

where $H(x)$ is the Heaviside step function, with parameters $r = 0.05$, $\sigma = 0.2$, $T = 1$, $S(0) = 100$,$K = 50$, investigate the use of importance sampling:

i. First, estimate the value without importance sampling.
How many samples are needed to obtain a value which is correct to within 10%? (i.e. the 3 standard deviation confidence limit corresponds to $\pm$ 10%).

ii. Second, try using importance sampling, adjusting the drift (i.e. changing the $(r-\frac{1}{2}\sigma^2)T$ term to a different constant) so that half of the samples are below the strike K, and the other half are above. Now how many samples are required to get the value correct to within 10%?

3. Look at the Matlab codes `lec5_weak.m` and `lec5_strong.m` which produced the plots in Lecture 5, and make sure that you understand what they are doing – ask if anything is unclear. Note that $g(e+\Delta e) \approx g(e) + \Delta e\, g'(e)$, so that if e is an estimate for $\mathbb{E}[f]$ with confidence interval $\pm 3\sigma/\sqrt{N}$ then $g(e)$ is an estimate for $g(\mathbb{E}[f])$ with confidence interval $\pm 3(\sigma/\sqrt{N})\, g'(e)$; this is used in `lec5_strong.m` to obtain a confidence interval for $\sqrt{\mathbb{E}[(\Delta S)^2]}$.

   Convert the codes to C++, python or R if you wish. (For the C++ code, I suggest you create an output file with the results data which you can then read into Matlab or python to do the plotting.)

   Modify `lec5_strong` for the Heston stochastic volatility model which is a coupled pair of SDEs:

$$\begin{aligned} \mathrm{d}S &= r\,S\,\mathrm{d}t + \sqrt{|v|}\,S\,\mathrm{d}W^{(1)}, \\ \mathrm{d}v &= \kappa\,(\theta - v)\,\mathrm{d}t + \xi\,\sqrt{|v|}\,\mathrm{d}W^{(2)}, \end{aligned}$$

   with $S(0) = 100, v(0) = 0.25, \theta = 0.25, \kappa = 2, \xi = 0.5$ over the time interval $[0, 1]$.

   The two driving Brownian motions are correlated so that

$$\mathbb{E}[\mathrm{d}W^{(1)}\mathrm{d}W^{(2)}] = -0.1\,\mathrm{d}t$$

   so the correlation matrix is

$$\Sigma = \begin{pmatrix} 1 & -0.1 \\ -0.1 & 1 \end{pmatrix}.$$

   There is no (easy) exact solution in this case so just plot the comparison between the h and $2h$ solutions.

   What is the order of strong convergence?

4. Apply MLMC to **a model of your own interest** or (if you are stuck) the problem suggested at the end.

   Mike Giles supports an extensive MLMC software webpage, and you can use any of this code to create your MLMC application.

- python groups: follow the link there to a bitbucket repository – the "opre" example has code to more-or-less replicate the results in Mike Giles's original MLMC paper (the original calculations were done in MATLAB using a different random number generator). If there are any problems in using the bitbucket repository try this zip file.
- C++ groups: the "mcqmc06" C++ code is for a different set of experiments in this paper.
- R groups: the link there takes you to the MLMC CRAN page; it also has an example which is more-or-less the same as in the original paper
- all groups: I suggest you run the codes, see the results you get, and then read through the codes in detail.

The routines like "mlmc" and "mlmc_test" are generic, the same for every application, and what the user has to write is the low-level "routine_l" code which computes the output correction on a particular MLMC level for the particular application of interest. For more details, see section 3 in Multilevel Monte Carlo methods, or Mike Giles's original MLMC paper.

Suggestion. Consider the 1D random PDE

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\kappa\,\frac{\mathrm{d}p}{\mathrm{d}x}\right) = 0$$

on the unit interval $0 < x < 1$, subject to $p(0)=0, p(1)=1$, where $u(x) \equiv \log \kappa(x)$ is Gaussian with covariance $\mathbb{E}[u(x)u(y)] = \frac{1}{4}\exp(-|x-y|)$.

The objective is to estimate the quantity

$$\mathbb{E}\left[\int_0^1 \kappa\,\frac{\mathrm{d}p}{\mathrm{d}x}\,\mathrm{d}x\right].$$

You can choose how to construct samples of u (Cholesky factorisation would be simplest), and how to approximate the 1D equation (e.g. finite difference or finite element method).