Monte Carlo Methods for Uncertainty Quantification: Practical 1

1. (a) Download from the course webpage the Matlab routines `ncf.m`, `ncfinv.m` and check what they do.

   (b) Generate 10^6 uniform random variables using `rand`, then convert them into 10^6 unit Normal variables using `ncfinv`. Check that they have the expected mean and variance.

   (c) Given a covariance matrix

   $$\Sigma = \begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{pmatrix}$$

   perform a Cholesky factorisation using Matlab function `chol(S,'lower')` to obtain a lower-triangular matrix L such that

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

   Use this matrix L to convert 3×10^6 independent unit Normals (generated using Matlab function `randn`) into 10^6 vectors of Normals with the desired covariance. Check that they have the expected mean and covariance.

   (d) Repeat the previous item using the PCA factorisation of Σ (using Matlab function `eig`).

2. 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) Using the Matlab `rand` function, write a Matlab program which computes

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

   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 (using my `ncf` or `ncfinv` functions) 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.

(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.

Following the Matlab code in lecture 1, 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?

3. For the case of Geometric Brownian Motion and a European call option, with parameters, $r = 0.05,\ \sigma = 0.2,\ T = 1,\ S_0 = 100,\ K = 100$, investigate the following forms of variance reduction:

(a) First, try antithetic variables using $\frac{1}{2}\left(f(Y) + f(-Y)\right)$ where Y is the $N(0,1)$ Normal random variable defined as $\Phi^{-1}(U)$, and U is the uniform $(0,1)$ random variable.

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

(b) Second, try using $\exp(-rT)\, S_T$ as a control variate, given that its expected value is S_0.

Again, how much variance reduction does this give?

4. Consider the case of a basket option based on 5 underlying assets, with $r=0.05,\ T=1$, and all 5 starting at $S_0=100$, and with the covariance matrix being

$$\Sigma = \sigma^2 \begin{pmatrix} 1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{pmatrix}$$

where $\sigma=0.2$. Consider a call option based on an equally-weighted average of the 5 assets, with strike $K=100$.

Investigate the variance reduction given by

(a) Latin Hypercube, using 1000 strata in each direction.
(You may find it helpful to use the Matlab `randperm` function.)

(b) Sobol quasi-random points, using 32 sets of scrambled points, and increasing the number of points within each set in powers of 2.
(As an alternative to Sobol points, you may like to try the lattice points generated by Dirk Nuyens' MATLAB codes on the webpage `http://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/`
)

In each case, check to see whether it makes a difference whether you use Cholesky or PCA factorisation.