Math G4077 – Homework Assignment 2* – Fall 2012 © Paul Feehan

1. Assignment references and sample code

1.1. **Reading assignments.** The primary texts and readings for the course are Glasserman, Joshi, Wilmott, and Achdou and Pironneau. You may find the suggested alternative readings helpful.

(a) Achdou and Pironneau, section 1.4.3.
(b) Brandimarte, sections 4.5.1 and 4.5.3.
(c) Clewlow and Strickland, sections 4.1–4.5.
(d) Glasserman, sections 1.1.3, 4.1.1, 4.2, A.1, and A.2.
(e) Jäckel, sections 2.4, 10.1, and 10.3.
(f) Joshi, chapters 2, 5, and 7; section 6.5.
(g) London, sections 2.5, 2.6, and 2.7
(h) Seydel, section 3.5.4.
(i) Tavella, chapter 5, pages 125-140.

1.2. **Sample code and libraries.**

(a) Main programs, `SimpleMCMainN.cpp`, $N = 1, \ldots, 4$, `StatsMain1.cpp` and `StatsMain2.cpp`, `EquityFXMain.cpp`, and their include files from `www.markjoshi.com/design/index.htm`.
(b) The sample code `bsma2.cpp` is a crude C++ program which uses the GNU Scientific Library to generate N(0,1) samples and provide the N(0,1) cumulative distribution function in an implementation of Monte Carlo and closed-form pricers for the European-style call option on an underlying asset modeled as geometric Brownian motion. The Monte Carlo code uses antithetic variance reduction and a computation timer.
(c) GNU Scientific Library Manual, `www.gnu.org/software/gsl/`; a high-quality numerical methods C/C++ library, `gsl` is freely available for Cygwin, Linux, or Unix.
(d) Numerical Recipes, `www.nrbook.com/a/bookcpdf.php`, with free C library; selected C++ code will be made available for download from Sakai.
(e) Ohio State University, Computational Physics 780.20, for sample Makefiles and GSL usage, `www.physics.ohio-state.edu/~ntg/780/`.

2. Programming and Written Assignments

Problem 2.1. Write a C++ program to price a European-style option using each of the methods described below; the option price style (call, barrier is up and out) and computational method are user-defined. For the program output test, assume that the stock price process is geometric Brownian motion with volatility $\sigma = 0.3$, initial asset price $S(0) = 100$, constant risk-free interest rate $r = 0.05$, dividend yield $d = 0.02$, option type call, strike $K = 110$, out barrier $U = 120$, and maturity $T = 1$ year; for the Monte Carlo simulations, use $J = 10,000$ paths, $I = 252$ time steps, and the Euler method of solution of $dS(t) = S(t)((r - d)\,dt + \sigma\,dW(t))$ to generate geometric Brownian motion sample paths, unless specified otherwise. When using control variates, choose $J_1 = 1,000$ as the number of pilot simulations used to estimate the optimal control coefficient, b^*, and $J_2 = 9,000$ as the number of pricing simulations used to estimate the desired option price.

Important: I provide specific **suggestions** for how to develop a C++ program using sample code provided by Joshi for the Monte Carlo and tree part of the course, **but** you may program the solutions however you wish using C++, **or** adapt MATLAB code of Brandimarte, **or** develop

*Last update: September 26, 2012 14:05

your own program from scratch however you wish, using C++, C, or MATLAB. (If you wish to use another language, please check with the teaching assistants first.) However you program the solutions, I recommend that that you use good object-oriented design (possible even using MATLAB) and avoid the temptation to simply "borrow" code from other sources; if you have not taken a course which covers object-oriented design, you may ignore this recommendation. The final projects will be designed so as to ensure that you need to write a substantial part of your own code. The remainder of the assignment refers to C++ but again, please **remember** the preceding comments.

Chapters 1-7 in Joshi are devoted to application of Monte Carlo to pricing European-style vanilla (call) and exotic options (that is, path dependent options, such as average, lookback, single or double barrier options). In coding your program, you may modify any one of the main programs (or their include files), `SimpleMCMainN.cpp`, $N = 1, \ldots, 5$, `StatsMain1.cpp` and `StatsMain2.cpp`, and `EquityFXMain.cpp`, authored by Joshi; if you wish to use one of them as a starting point, we recommend one of `SimpleMCMainN.cpp`, $N = 2, \ldots, 5$. However, when assembling your code submission file, please only include the classes and functions required; Joshi's code samples may include more functions than requested for the current assignment.

See section 7.3.1 in Shreve for a discussion of the up-and-out barrier option and Equations (7.3.19) or (7.3.20) for a closed-form formula for the price of an up-and-out barrier call option with payoff

$$(S(T) - K)^+ 1_{M(T) \leq U},$$

where $S(t)$ is the asset process, $M(t) = \max\{S(u) : 0 \leq u \leq t\}$ is its maximum process, and 1_A is one if the condition A is true and zero otherwise.

(a) Use the closed-form formula to compute the price for a European-style call option with an up-and-out barrier. Your submitted program should output the result as

```
Closed-form barrier option price =, run time =
```

(b) Use Monte Carlo simulation to compute the prices for a European-style call with an up-and-out barrier, the standard error, and the computation time. Your submitted program should output the result as

```
Monte Carlo barrier option price =, std error =, run time =
```

(c) Use Monte Carlo simulation with antithetic variance reduction to compute the prices for a European-style call with an up-and-out barrier, the standard error, and the computation time. Your submitted program should output the result as

```
Antithetic variates barrier option price =, std error =, run time
=
```

(d) Use Monte Carlo simulation with vanilla option control variates (with the same strikes) to compute the prices for a European-style call with an up-and-out barrier, the standard error, and the computation time. Your submitted program should output the result as

```
Control variates barrier option price =, std error =, run time =
```

(e) **Benchmarking.** Benchmark your results using Excel spreadsheets of Haug, Back, or Rouah-Vainberg, the MATLAB functions of Brandimarte or Mathworks' toolboxes.

(f) **Report.** Write a report (Word or LaTeX) which includes

- A short explanation of the algorithms and their implementation and an analysis of your results, including a comparison of your results with those obtain from an Excel-VBA spreadsheet or MATLAB algorithm.
- Answers to the questions below:
  - How effective is the vanilla option control variate for the up-and-out barrier call option when $U = 120, 200$, and $1,000$? What are your estimates for the control coefficient b^* and the correlation ρ?
  - How effective is the asset price control variate for the vanilla call option when $K = 20, 80, 100, 120$, and 200? What are your estimates for the control coefficient b^* and the correlation ρ?
  - How do the variance reduction methods affect computation time needed to achieve a specified accuracy?

3. Supplementary Notes

3.1. **Error analysis.** See Glasserman, section 1.1.3 for details. Let Y_i, $i = 1, \ldots, n$ be i.i.d. random variables on $(\Omega, \mathbb{P})$, having the same distribution as a random variable, Y. The sample mean, $\bar{Y} = n^{-1} \sum_{i=1}^{n} Y_i$, is an unbiased estimator for $\mathbb{E}[Y]$ with asymptotically valid $1 - \delta$ confidence interval

$$\bar{Y} \pm z_{\delta/2} \frac{s_Y}{\sqrt{n}},$$

where

$$s_Y := \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \bar{Y})^2},$$

is the sample standard deviation estimator and $\Phi(z_{\delta/2}) = 1 - \delta/2$, where $\Phi(x)$ is the cumulative distribution function for the standard normal random variable. When $z_{\delta/2} = 1$, then $\varepsilon := s_Y/\sqrt{n}$ is called the *standard error*. For a 95% confidence interval, one chooses $z_{\delta/2} \approx 1.96$.

3.2. **Antithetic variates.** See Glasserman, section 4.2 for details. Let $(Y_i, \tilde{Y}_i)$, $i = 1, \ldots, n$ be i.i.d. pairs of random variables on $(\Omega, \mathbb{P})$, where Y_i and $\tilde{Y}_i$ have the same distribution as random variable, Y. The pair $(Y, \tilde{Y})$ is *antithetic* if $Y = g(Z)$ and $\tilde{Y} = g(-Z)$, where Z is a standard normal random variable and $g(z)$ a deterministic function. For example, if $S(t)$ is geometric Brownian motion, then $(S(T), \tilde{S}(T))$ form an antithetic pair of random variables through the closed-form solution,

$$S(T) = S(0) \exp\left(\left(r - \frac{\sigma^2}{2}\right) T + \sigma\sqrt{T} Z\right),$$
$$S(T) = S(0) \exp\left(\left(r - \frac{\sigma^2}{2}\right) T - \sigma\sqrt{T} Z\right),$$

or through a numerical solution (such as the Euler algorithm) to the stochastic differential equation,

$$S(t_i + h) = S(t_i)\left(1 + rh + \sigma\sqrt{h} Z_{i+1}\right),$$
$$\tilde{S}(t_i + h) = S(t_i)\left(1 + rh - \sigma\sqrt{h} Z_{i+1}\right), \quad i = 1, \ldots, n.$$

The *antithetic variates estimator* is defined by

$$\hat{Y}_{\mathrm{AV}} := \frac{1}{n} \sum_{i=1}^{n} \left(\frac{Y_i + \tilde{Y}_i}{2}\right),$$

with asymptotically valid $1 - \delta$ confidence interval

$$\hat{Y}_{\mathrm{AV}} \pm z_{\delta/2} \frac{s_{\mathrm{AV}}}{\sqrt{n}},$$

where s_{AV} is the sample standard deviation of the n values $\frac{1}{2}(Y_i + \tilde{Y}_i)$ and $\Phi(z_{\delta/2}) = 1 - \delta/2$. One can show that one obtains a variance reduction if $\mathrm{Cov}[Y_i, \tilde{Y}_i] < 0$; this occurs when the option payoff – the output, Y_i – is a monotonic function of the Gaussian input driver, Z_i.

3.3. **Control variates.** See Glasserman, section 4.1 for details. Let (X_i, Y_i), $i = 1, \ldots, n$ be i.i.d. pairs of random variables on $(\Omega, \mathbb{P})$, having the same distribution as a pair of random variables, (X, Y). The (X_i, Y_i) will be outputs from n replications of a Monte Carlo simulation, where Y_i may be the discounted payoff of a derivative security evaluated on the ith simulated path and our objective is to estimate $\mathbb{E}[Y]$, while X_i have *known* (or easily estimatable) mean $\mathbb{E}[X]$. For any $b \in \mathbb{R}$, a *control variate estimator* for $\mathbb{E}[Y]$ is

$$\bar{Y}(b) := \bar{Y} - b(\bar{X} - \mathbb{E}[X]) = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - b(X_i - \mathbb{E}[X])\right),$$

where $\bar{Y} := n^{-1}\sum_{i=1}^{n} Y_i$ is the usual sample mean estimator and $Y_i(b) := Y_i - b(X_i - \mathbb{E}[X])$. The optimal coefficient,

$$b^* := \frac{\sigma_Y}{\sigma_X}\rho_{XY} = \frac{\mathrm{Cov}[X, Y]}{\mathrm{Var}[X]},$$

minimizes the variance

$$\mathrm{Var}[\bar{Y}(b)] = \frac{1}{n}\mathrm{Var}[\bar{Y}_i(b)] = \frac{1}{n}\left(\sigma_Y^2 - 2b\sigma_X\sigma_Y\rho_{XY} + b^2\sigma_X^2\right).$$

One finds that

$$\frac{\mathrm{Var}[\bar{Y}(b^*)]}{\mathrm{Var}[\bar{Y}]} = 1 - \rho_{XY}^2,$$

so we obtain a variance reduction when $\rho_{XY} \neq 0$ and the highest reduction when ρ_{XY}^2 is close to 1. In practice, σ_Y and ρ_{XY} are unknown and b^* must be estimated via $\hat{b}_n$, the slope of the least squares regression line through the points (X_i, Y_i), $i = 1, \ldots, n$,

$$\hat{b}_n := \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

We use $\bar{Y}(\hat{b}_n)$ as our control variate estimator, so the regression line passes through the point $(\bar{Y}(\hat{b}_n), \mathbb{E}[X])$. For large n, the correlation ρ_{XY} may be estimated via

$$\hat{\rho}_{XY} := \frac{\hat{b}_n}{\hat{\sigma}_X\hat{\sigma}_X} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2\sum_{i=1}^{n}(X_i - \bar{X})^2}},$$

using $\hat{\sigma}_X^2 := \sum_{i=1}^{n}(X_i - \bar{X})^2$ and similarly for $\hat{\sigma}_Y$. When n is small, the bias introduced by replacing b^* by $\hat{b}_n$ may become significant; this problem and approaches to mitigation are discussed in Glasserman, section 4.1.3.

An asymptotically valid $1 - \delta$ confidence interval for $\mathbb{E}[Y]$ is given by

$$\bar{Y}(\hat{b}_n) \pm z_{\delta/2}\frac{s(\hat{b}_n)}{\sqrt{n}},$$

where $\Phi(z_{\delta/2}) = 1 - \delta/2$ and

$$s(b) := \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i(b) - \bar{Y}(b)\right)^2}$$

is the sample standard deviation estimate for the standard deviation, $\sigma(b)$, of the $Y_i(b)$.

In practice, one performs a first set of n_1 pilot simulations to estimate b^* from the pairs (X_i, Y_i), $i = 1, \ldots, n_1$, and then a second set of n_2 pricing simulations to estimate $\mathbb{E}[Y]$ via the control variate estimator with coefficient $\hat{b}_{n_1}$; one should find that the $n_1 + n_2$ is much less than the

number of simulations, n, needed to achieve a given level of accuracy using simple Monte Carlo (without variance reduction). The mean, $\mathbb{E}[X]$, can be computed using a closed-form or another more accurate numerical method.

3.4. **Asian Options.** Asian options may be defined with continuous or discrete sampling and arithmetic or geometric averaging (see Back, section 8.9, or Haug, section 8.20):

$$\begin{aligned}
S_{AC}(T) &:= \frac{1}{T}\int_0^T S(u)\,du,\\
S_{GC}(T) &:= \exp\left(\frac{1}{T}\int_0^T \log S(u)\,du\right),\\
S_{AD}(T) &:= \frac{1}{N}\sum_{i=1}^{N} S(t_i),\\
S_{GC}(T) &:= \left(\prod_{i=1}^{N} S(t_i)\right)^{1/N} = \exp\left(\frac{1}{N}\sum_{i=1}^{N} S(t_i)\right),
\end{aligned}$$

where $t_1,\ldots,t_N$ are the asset price sampling dates and $0 = t_0 < t_1 < \cdots < t_N = T$. Assume $S(t)$ is geometric Brownian motion, $dS(t) = S(t)((r-q)\,dt + \sigma\,dW(t))$. There are no closed-form formulae for the arithmetic Asian options, but closed-form formulae for calls and puts on geometric averages can be derived using the method of Back, section 8.9. The price at time zero of a call on the geometric average option, continuously sampled over $[0,T]$, is

$$V_{CGC}(0) = e^{-rT}\mathbb{E}_{\mathbb{Q}}\left[(S_{GC}(T)-K)^+\right] = V(0)N(d_1) - e^{-rT}KN(d_2),$$

where d_1 and d_2 are defined as

$$d_1 = \frac{1}{\sigma_{\text{avg}}\sqrt{T}}\log\left(\frac{V(0)}{K} + \left(r + \frac{1}{2}\sigma_{\text{avg}}^2\right)T\right), \quad d_2 = d_1 - \sigma_{\text{avg}}\sqrt{T},$$

with volatility

$$\sigma_{\text{avg}} := \frac{\sigma}{\sqrt{3}}$$

and

$$V(0) := \exp\left(-\frac{1}{12}(6r+6q+\sigma^2)T\right)S(0).$$

The price at time zero of a call on the geometric average option, discretely sampled at $t_1,\ldots,t_N$, is

$$V_{CGD}(0) = e^{-rT}\mathbb{E}_{\mathbb{Q}}\left[(S_{GD}(T)-K)^+\right] = V(0)N(d_1) - e^{-rT}KN(d_2),$$

with d_1 and d_2 as above, using the volatility

$$\sigma_{\text{avg}} := \frac{\sigma}{N^{3/2}}\sqrt{\frac{1}{6}N(N+1)(2N+1)}$$

and

$$V(0) := e^{-rT}S(0)\exp\left(\frac{1}{2}(N+1)\nu\Delta t + \frac{1}{12N}(N+1)(2N+1)\sigma^2\Delta t\right).$$

where $\nu = r - q - \sigma^2/2$ and $\Delta t = t_{i+1} - t_i$, for $i = 0,\ldots,N-1$.