Math G4077 – Homework Assignment 4* – Fall 2012   © Paul Feehan

1. Reading Assignments 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. The primary reading assignments for this homework set are

- Glasserman, sections 4.3, 4.4, 4.6, and 7.1.
- Joshi, chapter 7.

You may find the following suggested complementary readings helpful:

- Brandimarte, sections 4.5.5, 4.5.6, 8.3.3, and 8.5.
- Clewlow and Strickland, section 4.6.
- Fusai and Roncoroni, section 1.3.3.
- Haug, section 8.4.2.
- Jäckel, sections 10.4, 10.5, 10.7 and chapter 11.
- Tavella, pages 140-152 and 155-164.

2. Programming and Written Assignments

Problem 2.1. This is *part two* of a two-part assignment to write a C++ program to price European-style, discretely-sampled, arithmetic and geometric Asian call (or put) options using the following test data and the methods described below.

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 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.

Write a C++ program (MATLAB users only may use the Brandimarte Sobol code with $d = 6$ rather than $d = 12$) to price European-style

- Vanilla call (or put) options,
- discretely-sampled, arithmetic Asian call (or put) options,
- discretely-sampled, geometric Asian call (or put) options

using the following test data and the methods described below.

- Spot process parameters:
  - Asset price process, $S(t)$ is geometric Brownian motion;
  - Volatility, $\sigma = 0.3$;
  - Initial asset price, $S(0) = 100$;
  - Dividend yield, $q = 0.02$;
- Discount curve parameters:
  - Constant risk-free interest rate, $r = 0.05$;

*Last update: October 19, 2012 7:46

- Payoff parameters:
  - Strike, $K = 110$;
  - Maturity, $T = 1$ year;
  - Asian option discrete sampling dates, $t_1, \ldots, t_d$, where $0 = t_0 < t_1 < \cdots < t_d = T$, $t_{i+1} - t_i = T/12$ for $i = 0, \ldots, d-1$, and $d = 12$;
  - Vanilla option sampling dates, $0 = t_0 < t_d = T$, where $d = 1$;
- Numerical method parameters:
  - Number of quasi or regular Monte Carlo simulation paths, $100,000$.

(a) Create your program by modifying Joshi's main program `EquityFXMain.cpp` and include files (only if necessary); `EquityFXMain.cpp` is configured to price an *arithmetic* Asian option (no closed-form formula), whereas the exercise also asks you to price a *geometric* Asian option (closed-form formula easily derived), and a *vanilla* option as a check on accuracy.

(b) Create a class (for example, coded as `GeometricAsianCF.h` and `GeometricAsianCF.cpp`) to compute the price of a European-style geometric Asian option, sampled at $t_1, \ldots, t_d$, using the closed-form formula in Homework Assignment 2.

(c) Create a class (for example, `BrownianBridgePath.h` and `BrownianBridgePath.cpp`) using the Brownian bridge algorithm to generate paths of Brownian motion at sampled at d discrete time points, $t_1, \ldots, t_d$, in $[0, T]$, for $0 = t_0 < t_1 < \cdots < t_d = T$.

(d) The Brownian motion path generation should allow for stratified sampling along $W(t_d)$.

(e) Modify Joshi's main program so vanilla as well as arithmetic and geometric Asian call options can be priced using regular Monte Carlo (with Park-Miller uniforms and inverse transform normals, with antithetic or terminal stratified sampling) and quasi Monte Carlo (d-dimensional Sobol sequence and inverse transform normals):

```
Closed-form vanilla call price =
MC vanilla call, Park-Miller uniforms, =
QMC vanilla call, Sobol sequence =

Closed-form geometric Asian call price =
MC geometric Asian call, Park-Miller uniforms, =
QMC geometric Asian call, Sobol sequence =

MC arithmetic Asian call, Park-Miller uniforms, =
QMC arithmetic Asian call, Sobol sequence =
```

(f) Program notes:
- Generation of paths of geometric Brownian motion is implemented in `ExoticBSEngine.h` and `ExoticBSEngine.cpp` and their include files.
- The vanilla option only requires simulation of the spot value at the maturity date, with no intermediate dates, and your program should be coded accordingly.

(g) **Benchmarking.** Benchmark your results using the Excel-VBA spreadsheets of Haug, Back, or Rouah-Vainberg, or the MATLAB functions of Brandimarte or Mathworks' toolboxes. You may also find that the following additional comparisons provide useful checks.

- Compare your results with those of the closed-form discretely sampled geometric Asian option from Kerry Back's spreadsheet (see `Discrete_Geom_Average_Price_Call` in the tab `Chapter8-Exotics`). Agreement should be exact.
- Compare the result from your code with those of the closed-form discretely sampled arithmetic Asian option from Espen Haug's spreadsheet (see the tabs `DiscreteAA` and `Curran` in the Chapter 4 Exotics spreadsheet `Asians`). Haug's algorithms use approximations, so agreement will be close but not exact.

(h) **Report.** Write a report (LaTeXpreferred, though Word is acceptable) which includes a short explanation of the algorithms and their implementation and an analysis of your results.

3. Practice Exercises and Review Questions

Problem 3.1. Explain what is meant by Latin hypercube sampling and its relation to low-discrepancy sequences.

Problem 3.2. Consult the given references and explain how to use stratified sampling or importance sampling or a combination to compute the following integrals or price the following products:

(a) Computation of π using importance (or stratified) sampling (Example 4.14 in Brandimarte) with
$$\pi = 4\int_0^1 \sqrt{1-x^2}.$$
(b) A deep out-of-the-money vanilla call using importance sampling (Brandimarte, pp. 266–267).
(c) A down-and-out barrier put option using importance sampling (Brandimarte, section 8.3.)
(d) An at-the-money European-style vanilla call using importance sampling (Tavella, pp. 143–149).
(e) Using terminal stratification and the Brownian bridge to price vanilla and path-dependent single asset, equity derivative products. See Glasserman, pp. 220–223, for an outline.
(f) Use the stratification of linear projections method (Glasserman, pp. 223-224) to price caplets and swaptions in the LIBOR market model. See Glasserman, section 3.7 and pp. 224–226 for an outline.
(g) A down-and-in digital option using importance sampling (Glasserman, Example 4.6.4).
(h) A discretely-sampled, arithmetic Asian call option, using importance sampling and a combination of importance and stratified sampling (Glasserman, section 4.6.2).

Problem 3.3. Explain some of the different methods of computing sensitivities ("Greeks") using Monte Carlo simulation.

Problem 3.4. Plot paths of Brownian motion over a time interval $0, 1]$ sampled at time points 2^{-d}, where $d \geq 2$ and observe convergence to a "typical", continuously sampled path of Brownian motion as $d \to \infty$.

4. Supplementary Notes

4.1. **Brownian bridge path generation.** The Brownian bridge path generation method is often used in conjunction with variance reduction methods, such as stratified sampling, and can be used with low-discrepancy sequences. Suppose that $W(t)$ is $\mathbb{R}$-valued Brownian motion on $(\Omega, \mathscr{F}, \mathbb{P}, \mathbb{F})$, where $\mathbb{F} = \{\mathscr{F}(t)\}_{t\geq 0}$ is a filtration for $W(t)$. Suppose that the values $W(s_1) = x_1, \dots, W(s_k) = x_k$ have already been generated and that we wish to generate $W(s)$, conditional on those k values, where $s \in (s_i, s_{i+1})$; this conditional distribution is the same as the distribution of $W(s)$, conditional on $W(s_i) = x_i$ and $W(s_{i+1}) = x_i$, and may be sampled using

$$W(s) = \frac{(s_{i+1} - s)x_i + (s - s_i)x_{i+1}}{s_{i+1} - s_i} + \sqrt{\frac{(s_{i+1} - s)(s - s_i)}{s_{i+1} - s_i}}Z,$$

where $Z \sim N(0, 1)$ is independent of $W(s_1), \dots, W(s_k)$. For details, see Glasserman, pp. 82–86, for scalar-valued Brownian motion and pp. 91–92 for vector-valued Brownian motion.

Given $(t_1, \dots, t_k)$ we can generate $(w_1, \dots, w_k)$ with the distribution of $(W(t_1), \dots, W(t_k))$ using the algorithm in Figure 3.2 in Glasserman, where we choose the smallest $2^m \geq k$:

```
• Generate $(Z_1, \dots, Z_{2^m}) \sim N(0, I)$;
• $h \leftarrow 2^m$, $j_{\max} \leftarrow 1$;
• $w_h \leftarrow \sqrt{t_h} Z_h$;
• $t_0 \leftarrow 0$, $w_0 \leftarrow 0$;
• for $k = 1, \dots, m$
    – $i_{\min} \leftarrow h/2$, $i \leftarrow i_{\min}$;
    – $\ell \leftarrow 0$, $r \leftarrow h$;
    – for $j = 1, \dots, j_{\max}$;
        * $a \leftarrow ((t_r - t_i)w_\ell + (t_i - t_\ell)w_r)/(t_r - t_\ell)$;
        * $b \leftarrow \sqrt{(t_r - t_\ell)(t_r - t_i)/(t_r - t_\ell)}$;
        * $w_i \leftarrow a + bZ_i$;
        * $i \leftarrow i + h$, $\ell \leftarrow \ell + h$, $r \leftarrow r + h$;
    – end;
    – $j_{\max} \leftarrow 2 * j_{\max}$;
    – $h \leftarrow i_{\min}$;
• end;
• return $(w_1, \dots, w_{2^m})$;
```

This is not necessarily an optimal solution, as $2^m - k$ samples will be discarded; see the next section for an alternative.

4.2. **Stratified sampling estimators.** Suppose Y is an $\mathbb{R}$-valued random variable on $(\Omega, \mathscr{F}, \mathbb{P})$ and that we wish to estimate $\mathbb{E}[Y]$. The random (or ordinary) sampling estimator for $\mathbb{E}[Y]$ is

$$\bar{Y} := \frac{1}{n}\sum_{i=1}^{n} Y_i,$$

where $Y_1, \dots, Y_n$ are iid with same distribution as Y.

4.2.1. *Proportional stratified sampling.* To construct the proportional stratified estimator for $\mathbb{E}[Y]$, we can use the first method of Glasserman, pp. 209–210, as described in Examples 4.3.1 and 4.3.2. If Y has distribution function $F : \mathbb{R} \to [0, 1]$ and $p_1, \dots, p_K$ are non-negative numbers with $\sum_{i=1}^{K} p_i = K$, define $a_0 := -\infty$,

$$a_i := F^{-1}(p_1 + \cdots + p_i), \quad , i = 1, \ldots, K,$$

and strata

$$A_1 = (a_0, a_1], A_2 = (a_1, a_2], \ldots, A_K = (a_{K-1}, a_K],$$

where $(a_{K-1}, a_K]$ is replaced by (a_{K-1}, ∞) if $a_K = \infty$. By construction, each stratum A_i has probability $\mathbb{P}\{Y \in A_i\} = p_i$ with respect to F. To generate a sample of Y conditional on $Y \in A_i$, we draw a sample $U \sim \text{Unif}[0, 1]$, set

$$V := a_{i-1} + (a_i - a_{i-1})U,$$

so $V \sim \text{Unif}[a_{i-1}, a_i]$, and observe that $F^{-1}(V)$ has the required conditional distribution. The proportional stratified sampling estimator for $\mathbb{E}[Y]$ is then

$$\hat{Y} := \frac{1}{n} \sum_{i=1}^{K} \sum_{j=1}^{n_i} Y_{ij},$$

where $n_i := np_i$ and Y_{ij} is the jth sample of Y for the stratum A_i. For a simple implementation, choose $p_i = 1/K$ and $n_i = n/K$ for $i = 1, \ldots, K$.

4.2.2. *Stratified sampling via an auxiliary stratification random variable.* See Glasserman, p. 210.

4.2.3. *Non-proportional stratified sampling.* See Glasserman, p. 211.

4.3. **Stratified sampling and error analysis.**

4.3.1. *Confidence intervals for stratified sampling estimators.* See Glasserman, p. 215–217.

4.3.2. *Optimal allocation for stratified sampling.* See Glasserman, p. 217–218.

4.3.3. *Stratified sampling estimators and their variance decomposition.* See Glasserman, p. 218–220.

4.4. **Applications of stratified sampling to option pricing.** The most common application uses terminal stratification.

4.4.1. *Terminal stratification.* If $S(t)$ is geometric Brownian motion and we wish to estimate an option price

$$\mathbb{E}[h(S(t_1), \ldots, S(t_m))],$$

where h is a discounted payoff function, we can use the technique of terminal stratification described in Glasserman, pp. 220–222, with $p_i = 1/K$ and $n_i = n/K$. We generate proportional stratified samples of $W(t_m)$, apply the Brownian bridge to construct the corresponding discrete-time paths, $W(t_i)$, for $i = 0, 1, \ldots, m$, and construct discete-time paths, $S(t_i)$, for $i = 0, 1, \ldots, m$, using

$$S(t_i) = S(t_0) \exp\left(\left(r - \frac{\sigma^2}{2}\right) + \sigma W(t_i)\right).$$

Rather than construct equiprobable strata over all possible terminal values, we may combine values of $S(t_m)$ which result in zero payoff into a single stratum and create a finer stratification of terminal values which potentially produce a non-zero payoff. This latter step is *not* required for the homework assignment.

We can use the algorithm of Glasserman, p. 221, to generate K paths of $W(t_1), \ldots, W(t_m)$ (and hence $S(t_1), \ldots, S(t_m)$), stratified along $W(t_m)$ (and hence $S(t_m)$) using the Brownian bridge:

```
• for $i = 1, \dots, K$,
    – generate $U \sim \text{Unif}[0,1]$;
    – $V \leftarrow (i - 1 + U)/K$;
    – $W(t_m) \leftarrow \sqrt{t_m}\Phi^{-1}(V)$;
    – for $j = 1, \dots, m-1$;
        ∗ generate $Z \sim N(0,1)$;
        ∗ $a \leftarrow \frac{t_m - t_j}{t_m - t_{j-1}}$, $b \leftarrow \frac{t_j - t_{j-1}}{t_m - t_{j-1}}$, $c \leftarrow \sqrt{\frac{(t_m - t_j)(t_j - t_{j-1})}{t_m - t_{j-1}}}$;
        ∗ $W(t_j) \leftarrow aW(t_{j-1}) + bW(t_m) + cZ$;
• return $(W(t_1), \dots, W(t_m))$;
• end;
```

Note that m is not assumed to be a power of 2. To obtain a corresponding path of geometric Brownian motion, one could use

$$\log S(t_i) = \log S(0) + \left(r - \frac{\sigma^2}{2}\right) t_i + \sigma W(t_i), \quad i = 1, \dots, m.$$

Alternatively, to better fit with the implementation in section 7.5 of Joshi, we could use

$$\log S(t_i) = \log S(t_{i-1}) + \left(r - \frac{\sigma^2}{2}\right)(t_i - t_{i-1}) + \sigma(W(t_i) - W(t_{i-1})), \quad i = 1, \dots, m,$$

and for the term $\sigma\sqrt{t_i - t_{i-1}}Z_i$ from Equations (7.2) and (7.3) in Joshi, replace the $Z_i \sim N(0,1)$ by $(W(t_i) - W(t_{i-1}))/\sqrt{t_i - t_{i-1}}$.