Math G4077 – Homework Assignment 1* – Fall 2012 © Paul Feehan

1. Assignment references and sample code

1.1. **Assignment references.** The primary texts and readings for this part of the course are in Glasserman and Joshi (if you use C++). You may find the suggested alternative readings helpful.

(a) Achdou and Pironneau, sections 1.1, 1.2, 1.3, and 1.4.
(b) Brandimarte, sections 4.2 and 4.4
(c) Glasserman, sections 3.1, 3.2.1, 3.2.3, 6.1, 6.2.1, and 6.2.2.
(d) Joshi, chapter 1.
(e) Seydel, sections 3.1, 3.2, 3.3, 3.5.1, and 3.5.2.
(f) Tavella, chapter 4, pages 77-93 and 100-102.

1.2. **Sample code and libraries.**

(a) The main C++ programs, `SimpleMCMainN.cpp`, where $N = 1, \ldots, 4$, and their include files available from `www.markjoshi.com/design/index.htm`.
(b) 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.
(c) Numerical Recipes, `www.nrbook.com/a/bookcpdf.php`, free C library; selected C++ code will be made available for download.
(d) Ohio State University, Computational Physics 780.20, for sample Makefiles and GSL usage, `www.physics.ohio-state.edu/$\sim$ntg/780/`.
(e) Brandimarte provides sample (procedural) MATLAB code for his book: `http://staff.polito.it/paolo.brandimarte/`

2. Programming and Written Assignments

Problem 2.1. Write a C++ program to price a European-style call or put option using each of the methods described below; the option price style 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 $K = 110$, and maturity $T = 1$ year.

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 use good object-oriented design (possible even using MATLAB) and avoid the temptation to simply "borrow" code from other sources. 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.

You may modify any one of the main programs (and their include files), `SimpleMCMainN.cpp`, $N = 1, \ldots, 4$, together with the code (`Normals.cpp`, `Normals.h`, `Random1.cpp`, and `Random1.h`), authored by Joshi.

The code `Random1.cpp` provides the Box-Muller method to generate samples from the univariate normal distribution using the `rand` function in <cstdlib>; see, for example, Glasserman, p.

*Last update: September 11, 2012 10:17

65, for details of the Box-Muller method. The code `Normals.cpp` provides the Hastings approximation to the cumulative normal distribution, as modified in Abramowitz and Stegun; see, for example, Glasserman, p. 67, for details of the Hastings approximation. Later in the course we shall discuss methods of generating sequences of pseudo-random and quasi-random numbers and samples from different probability distributions.

(a) Use the closed-form formulae to compute the prices for a European-style call and put. Your submitted program should output the result as

```
Option price using closed-form formula =
```

(b) Use the analytical solution, $S(T)$, to $dS(t) = S(t)((r-d)\,dt + \sigma\, dW(t))$ and Monte Carlo simulation to compute the prices for a European-style call and put. Choose one time step and $I = 10,000$ paths. Your submitted program should output the result as

```
Option price using single-step exact SDE solution =
```

(c) Use the Euler numerical solution to $dS(t) = S(t)((r-d)\,dt + \sigma\, dW(t))$ and Monte Carlo simulation to compute the prices for a European-style call and put. Choose I time steps using 252 time steps per year and $J = 10,000$ paths. Your submitted program should output the result as

```
Option price using Euler numerical solution of SDE for spot =
```

(d) Use the Euler numerical solution to $d\log S(t) = (r - d - \sigma^2/2)\,dt + \sigma\, dW(t)$, and Monte Carlo simulation to compute the prices for a European-style call and put. Choose I time steps using 252 time steps per year and $J = 10,000$ paths. Your submitted program should output the result as

```
Option price using Euler numerical solution of SDE for log spot =
```

(e) Use the Milstein numerical solution to $dS(t) = S(t)((r-d)\,dt + \sigma\, dW(t))$, and Monte Carlo simulation to compute the prices for a European-style call and put. Choose I time steps using 252 time steps per year and $J = 10,000$ paths. Your submitted program should output the result as

```
Option price using Milstein numerical solution of SDE for spot =
```

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

(g) **Report.** Write a report 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 (briefly describe the benchmarking algorithm).
- An answer to the question: Is it necessary to simulate entire paths, $\{S(t)\}_{t\in[0,T]}$, in order to compute the option price when the payoff is $(S(T) - K)^+$? What about when the payoff is that of a continuously monitored European-style Asian call option, $(\bar{S}(T) - K)^+$ where $\bar{S}(T) = T^{-1}\int_0^T S(t)\,dt$, or a discretely monitored European-style Asian call option, when $\bar{S}(T) = m^{-1}\sum_{i=1}^m S(t_i)$?