Math G4077 – Homework Assignment 6* – Fall 2012 © Paul Feehan

Practice or *advanced* problems are optional and are not graded: *practice problems* are intended as drill problems and aides to exam preparation while *advanced problems* are intended for students with additional mathematics background. Please consult the *Homework Submission Requirements* before commencing work on this assignment.

1. Reading Assignments and Sample Code

The primary reading assignments for this homework set are

- R. Cont and P. Tankov, *Financial modeling with jump processes*, section 12.4, Finite difference methods – theory and applications; for partial-integro differential equations (PIDEs) arising from stochastic models with jump processes.
- J. Thomas, *Numerical partial differential equations*, section 4.4.6, Parabolic equations with three space dimensions.
- P. Wilmott, *Quantitative Finance*, section 78.10, Jump conditions due to dividends; section 78.11, Path dependent options.

You may find the following suggested complementary readings helpful:

- Y. Achdou and O. Pironneau, chapter 4, Finite element method.
- Y. Achdou and O. Pironneau, chapter 6, American options.
- S. Crepey, sections 9–14, Finite difference and finite element methods for PIDEs.
- R. Cont and P. Tankov, section 12.6, Galerkin methods; for PIDEs arising from stochastic models with jump processes.
- R. Seydel, chapter 5, Finite element methods.
- R. Seydel, chapter 6, Pricing of exotic options.
- J. Thomas, section 4.4, Parabolic equations with two space dimensions.

Programming and Written Assignments

Problem 1.1. Write a C++ program to price European-style vanilla and barrier call options with the Crank-Nicolson finite difference scheme, using the following data.

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.

- Spot process parameters:
  - Asset price process, $S(t)$ is geometric Brownian motion;
  - Volatility, $\sigma = 0.3$;
  - Initial asset price, $S(0) = 100$;
  - Dividend yield, $q = 0.02$;

*Last update: November 3, 2012 9:56

- Discount curve parameters:
  - Constant risk-free interest rate, $r = 0.05$;
- Payoff parameters:
  - Call strike, $K = 110$;
  - Up-and-out barrier, $U = 120$;
  - Maturity, $T = 1$ year;
- Numerical method parameters:
  - $N_t = 252$/year time steps, $N_S = 50$ space intervals, minimum spot size $S_{\min} = 0.0$, and maximum spot size $S_{\max} = 200$ (without barrier).

(a) Implement a Crank-Nicolson finite difference class `CrankNicolsonFD` in the header and source files `CrankNicolsonFD.h` and `CrankNicolsonFD.cpp` for solving the general parabolic PDE at the bottom of page 1210 in Wilmott, using the finite difference scheme in Equations (78.2), where ν_1 and ν_2 are defined at the bottom of page 1228. For the Black-Scholes PDE with constant volatility, interest rate, and dividend yield, the coefficients a_i^k, b_i^k, and c_i^k may be identified by comparison of the PDEs at the bottom of page 1210.

(b) Implement the spatial boundary conditions ($i = 0$ and $i = I$) using section 78.3.1 (for vanilla options and barriers coinciding with a grid point) and section 78.3.2 in Wilmott for barriers which do not coincide with a grid point.

(c) Implement an LU decomposition class `LUDecomposition` to solve the matrix equation (78.4) as `LUDecomposition.h` and `LUDecomposition.cpp` using the method of section 78.3.5 in Wilmott.

(d) You may use use Joshi's `MJArray` class to define arrays, if you wish.

(e) Implement the Black-Scholes formulae for European-style vanilla calls using Joshi's files `BlackScholesFormulas.h`, `BlackScholesFormulas.cpp`, `Normals.h`, and `Normals.cpp`.

(f) Implement the closed-form formulae for European-style up-and-out barrier calls using code from Homework Assignment 2 as `ClosedFormUpOutCall.h` and `ClosedFormUpOutCall.cpp`.

(g) Write a *separate* main program, `CrankNicolsonFDMain.cpp`, which calls the above methods and produces the following outputs:

```
Closed-form European-style vanilla call price =
Crank-Nicolson FD European-style vanilla call price =
Closed-form European-style barrier call price =
Crank-Nicolson FD European-style barrier call price =
```

(h) **Program notes:** The program data should be included in the main program file, but not the class definition. Your program code archive file (`*.zip` or `*.tar.gz` **only**) must be complete and self-contained: include **all** required files.

(i) **Benchmarking.** Benchmark your results using the *Numerix* option pricer with the closest algorithms and numerical methods. If a Numerix option pricer is not available (you must explain why in your report), use the Excel-VBA spreadsheets of Haug, Back, or Rouah-Vainberg, or the MATLAB functions of Brandimarte or Mathworks' toolboxes.

(j) **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. How do your results vary with N_t, N_S, or $S_{\max}$? Are the stability and convergence criteria satisfied? Try a combination of step sizes to test robustness of the stability and convergence criteria.

- How does the explicit method compare with the Crank-Nicolson method in terms of stability and accuracy?
- A comparison of your results with those obtain from a Numerix, Excel-VBA spreadsheet, or MATLAB algorithm (briefly describe the benchmarking algorithm).

Practice Programming and Written Assignments

Problem 1.2. Rework the preceding problem using the explicit scheme when the options are now American-style.

Problem 1.3. Rework the preceding problem using the Crank-Nicolson scheme and the SOR method to solve the matrix equation when the options are now American-style.

Supplementary Notes

Please see the notes of Wilmott and Crepey on Sakai. Note that accuracy may be increased slightly by using linear interpolation, in the space dimension, of the option values at maturity when the initial spot value does not coincide with a grid point.