PROJECT Dr. Lin

MATH 6204 (8204)

Fall 2017

Due: Tuesday, December 12, 8:00 - 11:59 pm Name: _____

This project is to compute the values of European and American options on a dividend paying asset S that follows the geometric Brownian motion,

$$dS_t = (\mu - \delta)S_t dt + \sigma S_t dW_t,$$

where μ is mean return, δ is dividend yield, and σ is volatility (standard deviation). The riskless interest rate is denoted by r and it must be used to replace μ for the risk-neutral valuation of the options.

As is well-known, the time-t value of the option V satisfies the standard Black-Scholes equation,

$$\frac{\partial V(S,t)}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V(S,t)}{\partial S^2} + (r-\delta)S\frac{\partial V(S,t)}{\partial S} - rV(S,t) = 0.$$

This partial differential equation can reduce to its heat-equation equivalence,

$$\frac{\partial y(x,\tau)}{\partial \tau} = \frac{\partial^2 y(x,\tau)}{\partial x^2}, \quad x \equiv \ln(S/K).$$

You can refer to Topic 12 for the link between y and V and the link between t and τ as well as for other variable transformations. Note that variable V may refer to either an European call (put) or an American call (put), depending on the boundary conditions. For the European options, the boundary conditions and initial value conditions must be tackled carefully, and for American options, they must be treated as a linear complementarity problem.

For the project, the following parameter values are assumed:

- The price of the underlying asset today ($t = 0$): $S = 100$

- The strike price: $K = 100$

- The mean return: $\mu = 0.05$

- The risk-free interest: $r = 0.02$

- The dividend yield: $\delta = 0.01$

- Volatility: 0.6

- Maturity date: $T = 1$

- Present date: $t_0 = 0$

For this project, you are required to write a Python program and this program must include the driver file main.py and some other implementation files. You are expected to develop your Python program based on your independent efforts. If some part of your Python program is not written by your effort, you must indicate the sources of your code. If your Python program is partially or fully a duplicated copy from other students, then all these students who get involved one way or another will evenly share the project score (that is, each student's maximum score is at most 50 out of 100 if the project scores 100 out of 100).

Essentially, your Python program should have the capability to compute $V_c^{eur}(S = 100, t = 0)$, $V_p^{eur}(S = 100, t = 0)$, $V_c^{am}(S = 100, t = 0)$, and $V_p^{am}(S = 100, t = 0)$, where a call is index by "c" and a put by "p". For these computations, you will need to use the following methods:

1. Finite difference methods (FDMs):

(a) Explicit FDM (for both European and American options)
(b) Implicit FDM and Thomas algorithm (for European options)
(c) Implicit FDM and Brennan-Schwartz algorithm (for American options)
(d) Implicit FDM and SOR (for European options)
(e) Implicit FDM and Projected SOR (for American options)
(f) Crank-Nicholson and Thomas algorithm (for European options)
(g) Crank-Nicholson and Brennan-Schwartz algorithm (for American options)
(h) Crank-Nicholson FDM and SOR (for European options)
(i) Crank-Nicholson FDM and Projected SOR (for American options)

2. Monte Carlo simulation based on the explicit Euler method:

   (a) Monte Carlo integration of risk-neutral expectation (European options)
   (b) Regression method II (for American options)

3. Closed-form solution formulas (for European options)

Finite Difference Methods To apply the finite difference methods, you must discretize the heat equation into a linear system and then solve it using either direct or indirect (i.e. iterative) solvers. For this purpose, you must choose the finite x-domain $[x_{min} = -2.5, x_{max} = 2.5]$, the space step $\triangle x = 0.05$ and the time step $\triangle\tau = 0.00125$ to create a rectangular grid for the discretized linear system. This chosen mesh size leads to the magnitude of $\lambda = \frac{\triangle\tau}{\triangle x^2} = 0.5$, ensuring stability for the explicit FDM. As well, to solve the finite-difference systems with the SOR algorithm, you must the relaxation parameter at $\omega = 1.10$ and you must set the absolute-error tolerance at $tol = 10^{-6}$ so that the iteration can stop as tol falls below 10^{-6}:

$$\text{tol} \equiv \| w^{(k+1)} - w^{(k)} \| < 10^{-6}.$$

where w is the solution vector, k is the number of iterations, $\| w^{(k+1)} - w^{(k)} \|$ is the Euclidean norm.

Monte Carlo Simulation To apply Monte Carlo simulation based on the explicit Euler method, you can ignore the heat equation, and directly discretize the geometric Brownian motion of S_t and simulate the sample paths in the t-domain $[0, 1]$. To this end, you must set $\triangle t = 0.00125$ (that is, $M = 1/\triangle t = 800$) and the number of sample paths at $N = 500$. You are free to choose the seed number, though.

Note that in addition to computing the values of the European and American options, your Python program should be able to compute errors (in absolute value) between the approximation and the closed-form solution for the cases of European call and put options.

Also, note that the precision of each computed option value must have six decimals.

On the due date/time, you must email me with a package including:

1. **Summary your computed outputs and a short comment (10%)**: You must show a photocopy of your console outputs and you must type your summary and comments using font 12 (no more than two double-spaced pages) with your name on top of it. This can be either a Word file or a PDF file.

2. **Your zipped folded "project_FirstName_LastName"**

   Part I (10%, top of your main.py) – Your name, course title and number, and description of the project in terms of its purpose, model and algorithms. In addition, you must specify each individual file included in your zipped folder. Note that the project description must be reasonably detailed, so that the reader can understand it and know how to use your code to do the same computations with any other set of parameter values. If your project description cannot help the reader in this regard, that means the description is unclear!

Part II (80%, core of your Python code) – Your Python code should include remarks when necessary. It is critical to make sure that your Python program can run on command windows. Your score will also reflect whether or not your code is well structured and reader-friendly.

It is required that your code should be able to show tabulate console outputs with clear table headings and captions. The table form is up to your own design. Other than the way I used in class, you can also consider using the Pandas package for tabulating outputs (e.g., import pandas as pd). You are not allowed to use any other third-party packages for tabulating outputs.