IE523: Financial Computing
Fall, 2019
End-Term Programming Assignment: Verification of the Adjustment-Method for Pricing European Down-and-Out Continuous-, and European Down-and-Out Discrete-Barrier Options via Monte Carlo Simulation
Due Date: 13 December 2019 (On Compass)

Instructions

1. **If you worked with someone on a specific problem, make sure you tell me about it. If you do not do this and I notice a verbatim reproduction of material, I consider it as plagiarism. There can be serious consequences to this.**
2. **If you picked-up the solution from some source, make sure you mention it. If you do not do this and I notice a verbatim reproduction of material, I consider it as plagiarism. There can be serious consequences to this.**
3. **Each submitted document should be personalized. I expect to see (1) C++ code and header files (if any), (2) A two page write-up with information on your implementation that accompanies each submission by Midnight, 14 December, 2018 on Compass.**

There are two parts to this programming assignment that involves the simulation of price-paths using the material of Lesson 9. The first part is about pricing an *European Down-and-Out Continuous Barrier Option*, and the second part is about pricing an *European Down-and-Out Discrete Barrier Option*. You are in effect going to verify the adjustment-terms covered in section 1 of Lesson 7, and section 2 of Lesson 7. But this time, you are going to deal with the simulated price-paths assuming it follows the standard geometric Brownian motion model for asset-price. Let us review the details.

Review of the adjustment-term for the Continuous Barrier Option

According to Baldi, Caramellino and Iovino [**?**], suppose the initial asset-price is S_0, the final asset-price is S_T, the volatility is σ, and the barrier price is B, then the probability that an asset-price path that starts at S_0, ends up at S_T

in time T, breaches the barrier price of B is given by

$$p_c = \begin{cases} 1 & \text{if } S_0 \leq B \text{ or } S_T \leq B \\ e^{-\frac{2 \log \frac{S_0}{B} \log \frac{S_T}{B}}{\sigma^2 T}} & \text{otherwise.} \end{cases} \tag{1}$$

So, the probability that the asset price path that starts at S_0 and ends up in S_T in time T does not breach the barrier B is $1 - p_c$.

Baron-Adesi, Fusari and Theal [**?**] suggest that we just price the option without taking the Barrier into consideration, and then discount the price by multiplying it by $(1 - p_c)$. You will verify this assertion in the first-half of this programming assignment, and simultaneously, you will implicitly verify equation 1 shown above.

Review of the adjustment-term for the Discrete Barrier Option

A *Brownian bridge* is a continuous-time stochastic process $B(t)$, where

$$B(t) = X(t) - (t \times Z),$$

where Z is a unit-normal variate.

The Brownian bridge is very useful for the following task – suppose we have generated a number of points $X(0), X(1), X(2), \ldots X(n)$ of a Wiener process path by computer simulation. We now wish to interpolate between the samples. The solution is to use a Brownian bridge that is required to go through the sample points $X(0), X(1), X(2), \ldots X(n)$.

In general when $X(t_1) = a$ and $X(t_2) = b$, the **distribution** of $X(t_i)$ for $t_i \in (t_1, t_2)$ is normally distributed with mean μ_i

$$\mu_i = a + \frac{t_i - t_1}{t_2 - t_1} \times (b - a),$$

and variance

$$\sigma_i^2 = \frac{(t_i - t_1)(t_2 - t_i)}{t_2 - t_1}.$$

We have the initial price S_0, and we obtain the final price S_T at expiration T through a simulated price-path, we can check if $S_T > B$, the barrier-price. Following this, we can effectively compute the probability that the price path never touched any discrete barrier located at $t_1, t_2, \ldots, t_m (= T)$, as

$$p_d = \prod_{i=1}^{m-1} \left(1 - N\left(\frac{B - \mu_i}{\sigma_i}\right)\right) \tag{2}$$

Following what we did for the continuous barrier option, we price the discrete barrier option by ignoring the barrier entirely. Following this, we multiply the price by p_d shown in equation 1.

You will verify this claim in the second-part of the programming assignment. As before, verifying the adjustment-term is an indirect verification of the Brownian Bridge concepts introduced above.

What I want from you: First-Part

To get full points for the assignment, I want you to meet the following requirements[1]:

1. Your program should take the following command-line input (in the following order; this will make the grading-part easy):
   (a) Duration T,
   (b) Risk-free interest rate r,
   (c) Volatility σ,
   (d) Initial stock price S_0,
   (e) Strike Price K,
   (f) n, the number of trials/repetitions of the Monte Carlo simulations,
   (g) m, the number of sample-points in each sample price-path, and
   (h) Barrier Price B (assume the user will input a value of B such that $B < S_0$),
2. Your code will simulate a price-path of m-many equally spaced points. That is, your code will generate values for $S_0, S_1, S_2, \ldots, S_m (= S_T)$. You should exploit the "get-four-paths-for-the-price-of-one-path" method. If any $S_i \leq B$, the price path gets knocked-out, and gives you a zero-payoff. Otherwise, the payoff is $\max(0, S_T - K)$ for a call, and $\max(0, K - S_T)$ for a put. Present the average payoff over the n-many trials as the price of the option, which is obtained via explicit-simulation.
3. Using the same code, and just the values of S_0 and S_T, compute p_c as according to equation 1, and use a payoff $\max(0, S_T - K)(1 - p_c)$ for paths did not get knocked-out; all other paths have a payoff of zero. Present the average of these "adjusted" prices over the n-many trials as (an alternate) price of the option, which is obtained using just the initial- and final-price along with the adjustment-term.
4. Use the closed-form formulae in section 1.1 of lesson 7 to compute the theoretical price of the down-and-out option, and present it as the output too.

For the first-part, I am looking for an output that looks something like what is shown in figure 1. When all these three prices are approximately similar, we have effectively verified the formula shown in equation 1.

[1] If you do not follow my instruction, you will loose points unnecessarily.

```
profs-air:Debug sreenivas$ time ./Monte\ Carlo\ European\ Barrier\ Option 1 0.05 0.25 50 40 1000000 1000 20
---------------------------------
European Down-and-Out Continuous Barrier Options Pricing via Monte Carlo Simulation
Expiration Time (Years) = 1
Risk Free Interest Rate = 0.05
Volatility (%age of stock value) = 25
Initial Stock Price = 50
Strike Price = 40
Barrier Price = 20
Number of Trials = 1000000
Number of Divisions = 1000
---------------------------------
---------------------------------
The average Call Price by explicit simulation  = 12.7061
The call price using the (1-p)-adjustment term = 12.7061
Theoretical Call Price                         = 12.7063
---------------------------------
The average Put Price by explicit simulation  = 0.752408
The put price using the (1-p)-adjustment term = 0.752075
Theoretical Put Price                         = 0.751885
---------------------------------

real    2m51.925s
user    2m51.759s
sys     0m0.145s
profs-air:Debug sreenivas$
```

Figure 1: A sample output for the first part of the assignment.

What I want from you: Second-Part

To get full points for the assignment, I want you to meet the following requirements[2]:

1. Your program should take the following command-line input (in the following order; this will make the grading-part easy):
   (a) Duration T,
   (b) Risk-free interest rate r,
   (c) Volatility σ,
   (d) Initial stock price S_0,
   (e) Strike Price K,
   (f) n, the number of trials/repetitions of the Monte Carlo simulations.
   (g) m, the number of (equally-spaced) discrete barriers from 0 to T (note, the m-th barrier is at time T),
   (h) Barrier Price B (assume the user will input a value of B such that $B < S_0$),
2. Your code will simulate a price-path of m-many equally spaced points, which corresponds to the price of the stock at the location of each discrete

[2]If you do not follow my instruction, you will loose points unnecessarily.

barrier. Here too, you should exploit the "get-four-paths-for-the-price-of-one-path" method. If any $S_i \leq B$, the price path gets knocked-out, and gives you a zero-payoff. Otherwise, the payoff is $\max(0, S_T - K)$ for a call, and $\max(0, K - S_T)$ for a put. Present the average payoff as the price of the option over the n-many trials. This is the price obtained by the explicit simulation of the price-paths.

3. Using the same code, and just the values of S_0 and S_T, compute p_c as according to equation 1, and use a payoff $\max(0, S_T - K)p_d$, where p_d is given by the expression in equation 2, for paths did not get knocked-out; all other paths have a payoff of zero. Present the average of these "adjusted" prices as (an alternate) price of the option over the n-many simulations. This is the price of the option obtained by the brownian bride correction term.

For the second-part, I am looking for an output that looks something like what is shown in figure 2. When all these three prices are approximately similar,

```
profs-air:Debug sreenivas$ time ./Monte\ Carlo\ European\ Discrete\ Barrier\ Option 1 0.05 0.25 50 40 2000000 25 20
--------------------------------
European Down-and-Out Discrete Barrier Options Pricing via Monte Carlo Simulation
Expiration Time (Years) = 1
Risk Free Interest Rate = 0.05
Volatility (%age of stock value) = 25
Initial Stock Price = 50
Strike Price = 40
Barrier Price = 20
Number of Trials = 2000000
Number of Discrete Barriers = 25
--------------------------------
The average Call Price via explicit simulation of price paths            = 12.706
The average Call Price with Brownian-Bridge correction on the final price = 12.706
The average Put Price via explicit simulation of price paths             = 0.753049
The average Put Price with Brownian-Bridge correction on the final price  = 0.753592
--------------------------------

real    0m15.027s
user    0m15.004s
sys     0m0.017s
```

Figure 2: A sample output for the first part of the assignment.

we have effectively verified the formula shown in equation 2.

Everything that I asked for has to be implemented – no partial credit for partial implementations. I want these two programs uploaded on Compass by Midnight, December 13, 2019.

1. If your code works correctly on my data-sets (and this will require you to follow the input-format instructions rigorously), you get 100 points.

2. I will permit one back-and-forth iteration, if your submitted code does not

work[3]. That is, if I notice something wrong with your code, I will e-mail you right away. You have 24 hours to fix it and send it back to me.

(a) If your revised code works correctly you will get 80 points.

(b) Anything else, will get you zero points.

[3]It is not hard to verify things – get approximately the same prices for all variations of the problem!