FINM 32000: Homework 1

Due Wednesday January 27, 2020 at 11:59pm

Problem 1 of 2

Consider a particular stock index that is defined to have value equal to the price of a fixed basket of non-dividend-paying stocks. Suppose that it follows the Black-Scholes dynamics with $\sigma = 0.4$, and that the time-0 index level is $S_0 = 100$. Consider a three-month (=0.25 year) up-and-out European put, struck at 95, with a discretely-monitored knock-out barrier at 114, observed at times $0.02, 0.04, \ldots, 0.24$. That is, our option knocks out if and only if the index is at or above 114 at an observation time. Let the constant risk-free interest rate be $r = 0$.

The unit of time in this course will always be years, unless otherwise indicated.

(a) Write a Python function to price our option at time 0 using a trinomial tree. Some of the code is already provided for you.

Your code should choose the time step $\Delta t = T/N$ as suggested in class. We will want the barrier-monitoring times to be represented in the tree, preferably without introducing unequal time intervals anywhere, so we will want to choose N a multiple of 25. Your code should be able to accept any such N; a user who desires high accuracy can choose N large; a user who desires high speed can choose N small. For FINM 32000, please report a price using N chosen large enough that your output has converged, in your judgment (no proof needed), to within \$0.01 of the true price. In this example, $N = 100$ will not be sufficient.

Your code should choose the space step Δx such that the log of the barrier level is exactly halfway between consecutive log price levels of the tree. Subject to this constraint, choose Δx close to the recommended $\sigma\sqrt{3\Delta t}$ value. In other words, the constraint is that there exists an integer j such that $\log(114)$ is halfway between the jth and $(j+1)$th log-price levels:

$$\log S_0 + (j + 0.5)\Delta x = \log(114).$$

How should the integer j be chosen? Choose it in such a way that the Δx which satisfies the constraint is approximately $\sigma\sqrt{3\Delta t}$.

Why do we have this "halfway between" requirement? If you try instead putting $\log(114)$ *at* a log-price level, you will find the accuracy to be much worse than the "halfway between" procedure, for this discretely monitored barrier option.

(b) Consider an up-and-*in* put with the same terms. Specifically, this option has the same strike, expiry, barrier, and monitoring dates, but it pays at expiry the put payoff only if the index was at or above the 114 knock-*in* barrier at some monitoring date; otherwise, it pays nothing. Using your part (a) result, find the time-0 price of the up-and-in put.

(c) Consider a *continuously-monitored* barrier option paying at time $T = 0.25$, the amount

$$(95 - S_{0.25})^+ \mathbf{1}\big(\max_{0 \le t \le 0.25} S_t < 114\big),$$

where the indicator variable $\mathbf{1}(A) := \mathbf{1}_A := 1$ if event A occurs, 0 otherwise.

(c1) Is the time-0 price of the continuously-monitored barrier option greater than or smaller than the time-0 price of the discretely-monitored option in (a)? Justify briefly without doing any numerical calculations (one sentence is enough).

(c2) The continuously monitored barrier option can be replicated by a portfolio of T-expiry options, long 1 plain vanilla put struck at 95, and short α plain vanilla calls struck at 136.8.

The replication strategy is as follows. If S does not hit the barrier before time T, then simply collect the time-T payout of the 95 put, as desired. If S does hit the barrier, then at the time when S is *at the barrier*, the 1 unit of the vanilla put has value that *exactly cancels* the value of the $-\alpha$ units of the plain vanilla call; so at that time, we close out the portfolio positions, for a net payment of *zero*, as desired.

Solve analytically for the quantity α that makes this replication strategy valid, and find the time-0 value of the continuously-monitored barrier option. Do not use a tree.

Problem 1: Coding

Complete the coding of the function `barrier_trinom_pricer` in the provided file `hw1.ipynb`. The command `barrier_trinom_pricer(hw1dynamics, hw1contract, hw1tree)` must run properly, as it is currently written. Moreover, it should also still run properly, if you change the input parameters, such as `hw1tree.N` (which you will need to do, as $N = 100$ is insufficient for the desired level of accuracy), or other parameters, such as the volatility (therefore, the `0.4` volatility, and the other parameters should not be hard-coded into the function `barrier_trinom_pricer`.)

Do not modify the header of `barrier_trinom_pricer`. You may modify other lines in the file.

You do *not* need to make your code valid for contract parameters that would alter the contract's logic. In particular, you may assume that $H > S_0 > K$. Thus, you do not need to strive for maximum generality. But parameter perturbations that preserve the basic nature of the problem should run properly.

Problem 2 is at the end of this document

Problem 1: Discussion

- Note that the introductory paragraph of Problem 1 specifies the option *contract* and the underlying *dynamics*. The introductory paragraph says nothing about valuation *algorithm*.

  The knock-out dates, described in that paragraph, are features of the *contract*. They are written into the terms of the option. Whatever methodology/algorithm that a modeler might choose to *value* the contract (analytic approximation, or tree with 500 steps, or tree with 5000 steps, or Monte Carlo) has no impact on the specification of the *contract*, which dictates that the barrier observation times are $0.02, 0.04, \ldots, 0.24$.

- A useful diagnostic: test whether your code – with the barrier conditions *removed* – prices Europeans correctly, compared to Black-Scholes.

- When you introduce the barrier, one possible cause of error is code that fails to detect that you are at a monitoring date.

  Beware of code that tests whether two floating point numbers are equal by naively using `==`.

  Consider this example in Python:

```
In [1]: 0.14 / 0.02 == 7
Out[1]: False
```

  My computer thinks that .14/.02 and 7 are *not* equal.

  That's because it calculates .14/.02 and gets 7.000000000000001

  Floating point arithmetic should not be assumed to be exact. If you need to test whether two floating point numbers `x` and `y` are equal, then instead of using `==`, it would be better to test whether `abs(x-y)` is smaller than some tolerance.

Problem 2

Read page L2.12.

Suppose that a non-dividend-paying stock has dynamics

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma(t)S_t\mathrm{d}W_t, \tag{1}$$

where W is Brownian motion under risk-neutral probabilities, and where the time-dependent but *non-random* volatility function $\sigma : [0, T] \to \mathbb{R}$ is piecewise continuous and sufficiently integrable. Let's relate this instantaneous volatility function σ to the Black-Scholes implied volatility σ_{imp}.

(a) Are the dynamics (1) capable of generating a non-constant (with respect to T) term-structure of implied volatility? Are they capable of generating an implied volatility skew (non-constant with respect to K)? Explain briefly.

(b) Let $S_0 = 100$ and $r = 0.05$. At time 0, you observe the prices of at-the-money (this means $K = 100$) European calls at 0.1-year, 0.2-year, and 0.5-year expiries to be 5.25, 7.25, and 9.5, respectively. First find the *B-S implied* volatilities of the three options, by completing the coding of the `IVofCall` function.

Then find (calibrate!) a time-varying *local* volatility function $\sigma : [0, 0.5] \to \mathbb{R}$ consistent with these option prices. A step function suffices (but other answers are also acceptable).

(c) Consistently with your local volatility function σ from part (d), find the time-0 price of an at-the-money European call with expiry 0.4. Do not use a tree.