The Cooper Union Department of Electrical Engineering
Prof. Fred L. Fontaine
ECE478 Financial Signal Processing
Problem Set II: Binomial Asset Pricing Model
October 20, 2022

Write code to simulate the BAPM and perform analysis as suggested. Notation and so forth follows as given in Shreve, *Stochastic Calculus for Finance vol. 1.*
I will describe what I expect in terms of general symbols, first. Then you will run some tests with specific suggested values.
Your simulation should take r, d, u as given, and constant at all time steps. Building in error detection to ensure these satisfy the no-arbitrage constraint is optional. Might as well take $S_0 = 1$. We will assume the coin toss distribution is constant over time, but the particular (p, q) values can be variable. Two special cases are the "actual" probabilities (p_0, q_0), and the risk-neutral measure $(\tilde{p}, \tilde{q})$. Let $\mathcal{F}_n$ be the σ-algebra generated by the first n tosses, which covers the time span $\{0 \leq k \leq n\}$. Every stochastic process we consider here is adapted to this filtration.
Let $\vec{\omega}_n = (\omega_1 \cdots \omega_n) \in \{H, T\}^n$ denote a particular 'path' through the first n tosses. Note that, because $u \neq d$ here, $\vec{\omega}_n$ uniquely determines $(S_0 = 1, S_1, \cdots, S_n)$ and conversely. That means any other security in our market model is actually a derivative of S_n, specifically:

$$V_N = v_N(S_0, S_1, \cdots, S_N)$$

where v_N is a deterministic function. This general form is *path dependent,* as we can also view $V_N = V_N(\vec{\omega}_N)$. As a special case, if $V_N = v_N(S_N)$ depends on the final value only, it is called *path independent.* Whereas there are 2^N paths $\{\vec{\omega}_N\}$ of length N, there are only $N+1$ possible values for $\{S_N\}$, so path independent derivatives are much easier to simulate. The notes provide a precise distribution for S_N in terms of p (essentially a transformed binomial distribution). It should also be straightforward to express $P(\vec{\omega}_N)$ in terms of p, if we need to consider a full path instead of just the terminal state. In any case, please read carefully below– when the object of interest is path independent, take advantage of that to reduce computational complexity.
Your model should cover the time steps $\{0 \leq n \leq N\}$. When you perform a Monte Carlo simulation, M is the number of samples used (i.e., for purposes of Monte Carlo, you repeat the experiment M times and average over those results).
In what follows, for any stochastic process X_n, the *discounted* process is $\tilde{X}_n = \frac{1}{(1+r)^n} X_n$.

1. **Exact simulation:** Assume we have code to compute the payout function $V(S_N)$ for a path-independent derivative.

   (a) Write code to compute $E_p\left(\tilde{V}_N\right)$ for given p. Later, when you set $p = \tilde{p}$, this will give you the actual price V_0 of the derivative at $t = 0$. Note that as a path-independent derivative, you can directly use the known distribution of S_N (and specifically there are only $N+1$ values). [If it was path-dependent, you would need to average over 2^N paths using the direct probabilities $P(\vec{\omega}_N)$]

(b) Write code to generate one step of the replicating portfolio. If at time n there are Δ_n shares of the stock, and wealth X_n, then the amount $M_n = X_n - \Delta_n S_n$ is held in the money market. From $n \longrightarrow n+1$, the stock price changes from S_n to S_{n+1}, and the amount in the money market grows to $(1+r) M_n$. Thus, at time $n+1$, the *wealth equation* states:

$$X_{n+1} = \Delta_n S_{n+1} + (1+r)(X_n - \Delta_n S_n)$$

For this to be a replicating portfolio, $X_n = V_n$, the price of the derivative at time n, where at $n = N$ this should match $V_N(S_N)$ exactly. Give $\vec{\omega}_n$, $0 \le n \le N-1$, there are two possibilities for $\vec{\omega}_{n+1}$, namely $(\vec{\omega}_n H)$ and $(\vec{\omega}_n T)$. Assuming $V_{n+1}(\vec{\omega}_n H)$, $V_{n+1}(\vec{\omega}_n T)$ are both known, and of course $S_{n+1}(\vec{\omega}_n H), S_{n+1}(\vec{\omega}_n T), S_n(\vec{\omega}_n)$ are all known, with given r, the wealth equation yields two linear equations in the unknowns $\Delta_n(\vec{\omega}_n), X_n = V_n(\vec{\omega}_n)$.

(c) Extend your one-step routine in part **(b),** running it recursively backwards, to derive $X_0 = V_0$ and:

$$\{\Delta_n(\vec{\omega}_n)\}_{\text{all } \vec{\omega}_n, \text{ for } 0 \le n \le N-1}$$

1. The notation used above is general, and implies Δ_n is potentially path dependent. However, if V_N is path independent, is Δ_n path independent?
2. Write code to compute X_0 and all $\{\Delta_n\}$. The code should be for the case V_N is path *independent.* Thus, if Δ_n is path dependent, you should compute 2^n values of Δ_n for each n (the number of paths); but if path independent, compute $n+1$ values for each n (the number of S_n values). **Remark:** If you are not sure, try the path dependent case, and compare values, see if you always get the same result! **Remark:** Whether path dependent or not, there are a variable number of Δ_n values for each n. I leave it to you to decide the best approach to handle this in the code, except to point out it is best not to use separate variable names for each $\vec{n}$ (i.e., *delta0, delta1, delta2, ...);* instead you should pack ALL the Δ_n's into a single data structure!

(d) Now let us take a specific example to test your code. Take $r = 0.05$, $u = 1.1$, $d = 1.01$, $N = 5$, and the derivative a European call option with strike price $K = (1+r)^N S_0$.

1. Select two probabilities p_1, p_2, one greater than $\tilde{p}$, the other lower. Compute $E_p\left(\tilde{S}_N\right)$ and $E_p\left(\tilde{V}_N\right)$ for $p_1, p_2, \tilde{p}$. In one of these cases, you should get the "correct" values for S_0 and V_0 (V_0 to be verified below a different way): which case? For the "incorrect" values, are you observing a risk premium?
2. Now compute the complete set of replicating portfolio values $\{\Delta_n(\vec{\omega}_n)\}$ and V_0. Verify that your value of V_0 matches what you got via expectation. **Question:** Do you have to do any short selling of the stock or borrowing from the money market along the way?

2. **Monte Carlo simulation:** Next is to deal with the case $N = 100$. Also here, I want your code to be written for the more general *path dependent case,* even though the specific derivative we will test it on is path independent. Since the number paths

grow exponentially over time, including 2^{100} paths over the full time span, this is definitely a candidate for Monte Carlo methods. Let me first describe the code you should create. For fixed m, and an adapted stochastic process X_n, we want to compute $Y_n = E_{\tilde{p}}\left(X_{n+m}|\mathcal{F}_n\right)$. What this means is $Y_n = Y_n\left(\vec{\omega}_n\right)$, a function of the path associated with the first n tosses. So assume that is given (to be specific, compute this for a **single** value of $\vec{\omega}_n$ that can be prescribed as an input). You will generate M random paths $(\omega_{n+1}\cdots\omega_{n+m})$ according to the distribution, compute $X_{n+m}\left(\vec{\omega}_n\omega_{n+1}\cdots\omega_{n+m}\right)$ for each, and average over them to get Y_n. The details of computing X_{n+m} should be contained in a function that is called by the Monte Carlo "wrapper".

(a) As a first step, let us check the underlying behavior of your Monte Carlo code. Go back to the previous case, with $N = 5$ and r, u, d as given. Taking $M = 1, 5, 10, 32$, estimate S_0 from S_N, and V_0 from V_N. Note that you are not going through every path systematically! You are generating paths independently, and so it is possible (especially in the low order case) the same path will occur multiple times, so even with $M = 32$ there is no guarantee you will get all paths! Hence your S_0, V_0 estimate will not be exact. The idea is to see how close your estimates are for various M.

(b) Now take the case $N = 100$. In this case, we should change some of the parameters. Take $r = 10^{-3}$ (so over 100 steps there is a total return of about 10%), $u = 1+5\times10^{-3}$, $d = 1+10^{-4}$. Take different values of M, starting small so your computer doesn't take too long to run, and increasing it somewhat. We want to hopefully see an effect of increasing M, without forcing you to run simulations overnight! First, as a check, we know we should get $S_0 = E_{\tilde{p}}\left(\tilde{S}_N\right)$ and fortunately we know $S_0 = 1$. See how this works for various M. Next, again let $V_N\left(S_N\right)$ be a European call with $K = (1+r)^N S_0$ Compute $V_0 = E_{\tilde{p}}\left(\tilde{V}_N\right)$ for the same M as before.

(c) Continuing with the case $N = 100$ above. Take 5 random paths $\vec{\omega}_{10}$ of length 10 using $p = 0.9\tilde{p}$ (i.e., we are using the "actual" probabilities). For each path, compute $S_{10}\left(\vec{\omega}_{10}\right)$; then use a Monte Carlo approach to estimate $S_{10}\left(\vec{\omega}_{10}\right)$ and $V_{10}\left(\vec{\omega}_{10}\right)$; try different values of M, but for each choice of M you should do this for all of your 5 random paths. You can use how close the results are to S_{10} (which you know) to see how well the Monte Carlo method is working. The other thing to investigate: for a fixed path choice $\vec{\omega}_{10}$ and value M, repeat the Monte Carlo experiment multiple times and compute (1)the sample mean of S_{10}, V_{10} you are getting; and (2)the sample variance of these values. This provides additional insight to how well the Monte Carlo method is working. **Comment!**

(d) Repeat parts **(b),(c)** above for the (path dependent) lookback option:

$$V_N = \max_{0\le n\le N}\,(1+r)^{N-n}\,S_n - S_N$$

The first term is the maximum value of the stock price viewed under future valuing (i.e., projected to time N). This is where the value of a Monte Carlo approach becomes clear! **Remark:** Although V_N is path dependent, there is still a possible reduction. If:

$$M_n = \max_{0 \le k \le n} (1+r)^{N-n} S_n$$

then the dependency of V_N on a path $(\vec{\omega}_n, \omega_{n+1}, \cdots, \omega_N)$ takes the form:

$$V_N(\vec{\omega}_n, \omega_{n+1}, \cdots, \omega_{n+M}) = V_N(M_n, S_n, \omega_{n+1}, \cdots \omega_N)$$

That in, the only information required from the first n tosses, in order to extend out to N to evaluate V_N, are the values of M_n, S_n (they serve the role of *state variables* in the model). So, in repeating part **(c),** once you have $\vec{\omega}_{10}$ in hand, compute M_{10} and S_{10} and proceed. Check this. As a hint, first note $\omega_{n+1}, \cdots, \omega_N$ are not enough to find $S_{n+1}, \cdots, S_N$, but if we also know S_n, then it is enough; then, why is $M_n, S_{n+1}, \cdots, S_N$ sufficient to find M_N?