FE545 - Final Exam

Author: Sid Bhatia

Date: May 3th, 2024

Pledge: I pledge my honor that I have abided by the Stevens Honor System.

Professor: Steve Yang

Background

Tree-based methods can be used for obtaining option prices, which are especially popular for pricing American options. Binomial (BAPM/BOPM), trinomial (TAPM/TOPM), and random tree methods (RTM) can be used to price many options, including plain vanilla options, and also exotic options such as barrier options, digital options, Asian options, and others.

We will be pricing American options using TAPM/RT methods via design principles in C++.

Pricing a derivative security entails calculating the **expected value of its payoff**. This reduces, in principle, to a problem of *numerical integration*; but in practice this calculation is often difficult for high-dimensional pricing problems.

The binomial options pricing model (BOPM) approach has been widely used since it is able to handle a variety of conditions for which other models cannot easily be applied. This mainly since the BOPM is based on the description of an underlying instrument over a period of time rather than a single point. Therefore, it can be used to price Americans that are exercisable $\forall\, t \in [0, T]$.

The TOPM, proposed by **Phelim Boyle** in '86, is (considered to be) more accurate than BOPM, producing the same results w/fewer steps.

Broadie and **Glasserman** ('97) proposed the simulated random tree to price Americans, deriving the upper/lower bounds; this combo makes it easier to measure & control error as computational effort increases $(m \to \infty)$. The main drawback of RTM is *computational requirements* grow exponentially with m, # of exercise dates, applicable only when $m < \text{big number}$. For $m = \text{small number}$, RTM works well & shows theme of **mananging scores** of **high/low bias**.

The typical simulation to Euro pricing is use sims to $\approx$ the expectation:

$$P = e^{-rT}\mathbb{E}^{Q}[f(S_T)] \tag{1}$$

where $f(S_T)$ is the payoff function at maturity T. For a call, $f(S_T) = (S_T - K)_+$. r denotes the risk-free rate, K is the strike, and S_r is the **terminal stock price**.

For Americans, the dilemma is as follows:

$$P = \max_{\tau}\{e^{-r\tau}(S_\tau - K)_+\},\ \forall\tau \le T \tag{2}$$

We discretize this problem where $\tau \in \mathcal{P}$, with $\mathcal{P} = \{t_0, t_1, \ldots, t_d\}$ such that $t_0 < t_1 < \cdots < t_d = T$. For an American option, we simulate a path of asset prices $(S_0, S_1, \ldots, S_T)$. Let $i \in \{1, \ldots, d\}$ correspond to intermediate times t_i in $\mathcal{P}$.

To calculate the **discounted value** of an option for a given simulation path, you first determine the payoff for each path at maturity and then average these results over many simulations. This process helps estimate the expected payoff under stochastic conditions. How do we compute the value along each path?

Broadie and **Glasserman** (1997) developed a stochastic method to estimate lower and upper bounds for American options. Let $\tilde{h}_i$ denote the **payoff function** for exercise at t_i, depending on i. Let $\tilde{V}_i(x)$ denote the value of the option at t_i given $X_i = x_i$ (assuming the option has **not been exercised**).

Why do we care? We have an interest in $\tilde{V}_0(X_0)$, which is recursively defined as follows:

$$\tilde{V}_m(x) = \tilde{h}_m(x) \tag{3}$$

$$\tilde{V}_{i-1}(x) = \max\{\tilde{h}_{i-1}(x),\, \mathbb{E}[D_{i-1,i}\tilde{V}_i(X_i) \mid X_{i-1} = x]\} \tag{4}$$

For each i from 1 to $m-1$, we introduce the notation $D_{i-1,i}$ for the discount factor from time t_{i-1} to t_i. This ensures that at the $(i-1)$-th exercise date, the option value (OV) is the maximum of the **immediate exercise value** and the **expected present value** of continuing. At expiration, the option value is given by the payoff function $\tilde{h}_m$.

As the name indicates, the random tree method (define RTM here) is based on simulating a *tree of paths* of the underlying **Markov chain** $\{X_0, X_1, \ldots, X_m\}$. Let's assume we fix a *branching parameter* $b \geq 2$, where $b \in \mathbb{Z}^+$ (for simplicity).

From the initial state X_0, simulate b independent successor states $\{X_1^1, \ldots, X_1^b\}$, each following the same probability distribution (**law**) as X_1. From each X_1^i, simulate b independent successors $X_2^{i1}, \ldots, X_2^{ib}$, each following the **conditional law** of X_2 given $X_1 = X_1^i$.

From each $X_2^{i_1 i_2}$, generate b successors $X_3^{i_1 i_2 1}, \ldots, X_3^{i_1 i_2 b}$, and so on up to X_m.

We denote a **generic node** in the tree at time step i by $X_i^{j_1 j_2 \cdots j_i}$. This notation indicates that this node is reached by following the j_1-th branch out of X_0, the j_2-th branch out of the subsequent node, and so forth, reflecting the path taken through the branching tree.

At all terminal nodes of the tree, the estimator is set equal to the payoff at that node:

$$\hat{v}_m^{j_1\cdots j_m} = h_m(X_m^{j_1\cdots j_m}) \tag{5}$$

This means that at the final time step, the value of the option is **exactly equal** to its **payoff**, as there are no subsequent decisions to make.

At each node and at each branching step within the tree, the decision is calculated based on the **potential future values** versus the **immediate payoff**:

$$\hat{v}_{ik}^{j_1j_2\cdots j_i} = \begin{cases} h_i(X_i^{j_1j_2\cdots j_i}) & \text{if } \frac{1}{b}\sum_{j=1}^{b} \hat{v}_{i+1}^{j_1j_2\cdots j_ij} \leq h_i(X_i^{j_1j_2\cdots j_i}) \\ \hat{v}_{i+1}^{j_1j_2\cdots j_ik} & \text{otherwise} \end{cases} \tag{6}$$

This evaluates whether to exercise the option now or *continue* to the next step, based on expected values calculated from future possible states.

Next, the estimator for each node is computed as the average of the estimators from each possible path branching from this node:

$$\hat{v}_i^{j_1\cdots j_i} = \frac{1}{b}\sum_{k=1}^{b} \hat{v}_{i,k}^{j_1\cdots j_i} \tag{7}$$

This calculation effectively averages the outcomes from all *potential future states* accessible from the current node, **weighting each equally**. This average is crucial for determining the expected value of the option at each node, considering all possible future paths.