FE545 Design, Patterns and Derivatives Pricing

More on Trees

Steve Yang

Stevens Institute of Technology

steve.yang@stevens.edu

03/26/2024

Overview

Price Options using Trinomial Tree

- The trinomial option pricing model is an option pricing model incorporating three possible values that an underlying asset can have in one time period.
- The three possible values the underlying asset can have in a time period may be greater than, the same as, or less than the current value.
- The trinomial option pricing model, proposed by Phelim Boyle in 1986, is considered to be more accurate than the binomial model, and will compute the same results, but in fewer steps.
- Under the trinomial method, the underlying stock price is modeled as a recombining tree, where, at each node the price has three possible paths: an up, down and stable or middle path. We will use a trinomial tree model defined by

$$S(t+\Delta t) = \begin{cases} S(t)u & \text{with probability} \quad p_u \\ S(t) & \text{with probability} \quad 1-p_u-p_d \\ S(t)d & \text{with probability} \quad p_d \end{cases} \tag{1}$$

Price Options using Trinomial Tree

- Condition (2) is a standard equilibrium or no arbitrage condition: it states that the average return from the asset should be equal to the risk free-return.

$$\begin{aligned} \mathbb{E}[S(t_{i+1})|S(t_i)] &= e^{r\Delta t} S(t_i) \\ Var[S(t_{i+1})|S(t_i)] &= \Delta t S(t_i)^2 \sigma^2 + \mathcal{O}(\Delta t) \end{aligned} \tag{2}$$

- It can be re-written in the following explicit form:

$$1 - p_u - p_d + p_u u + p_d d = e^{r\Delta t} \tag{3}$$

Price Options using Trinomial Tree

- Conditions (2, 5) impose two constraints on 4 parameters of the tree.
- An extra constraint comes from the requirement that the size of the upward jump is the reciprocal of the size of the downward jump,

$$ud = 1 \tag{4}$$

- While this condition is not always used for a trinomial tree construction, it greatly simplifies the complexity of the numerical scheme: it leads to a recombining tree, which has the number of nodes growing polynomially with the number of levels rather than exponentially.
- Given the knowledge of jump sizes u, d and the transition probabilities p_u, p_d it is now possible to find the value of the underlying asset, S, for any sequence of price movements.

Price Options using Trinomial Tree

- Let us define the number of up, down and middle jumps as N_u, N_d, N_m, respectively, and so the value of the underlying share price at node j for time i is given by

$$S_{i,j} = u^{N_u} d^{N_d} S(t_0), \text{ where } N_u + N_d + N_m = N \tag{5}$$

- We have imposed three constraints (1, 2 and 4) on four parameters u, d, p_u and p_m. As a result there exists a family of trinomial tree models.
- In the project we consider the following popular representative of the family: its jump sizes are

$$\begin{aligned} u &= e^{\sigma\sqrt{2\Delta t}} \\ d &= e^{-\sigma\sqrt{2\Delta t}} = \frac{1}{u} \\ m &= 1 \end{aligned} \tag{6}$$

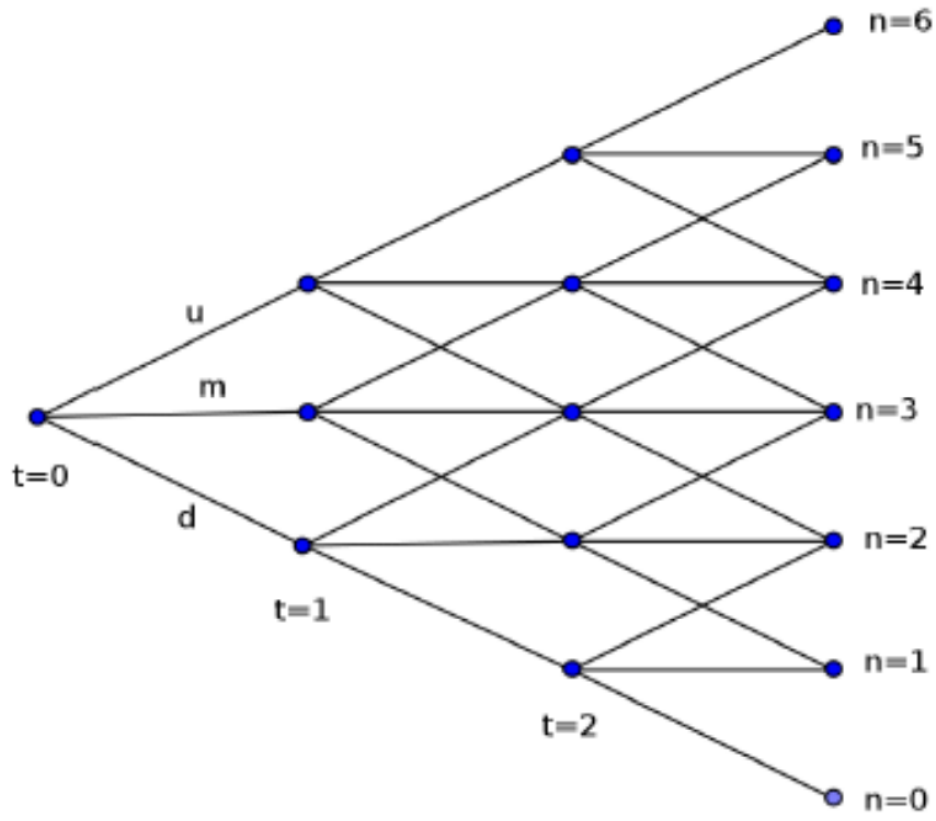

Figure: Trinomial Tree Option Pricing Example.

Price Options using Trinomial Tree

- The corresponding probabilities are:

$$p_u = \left(\frac{e^{(r-q)\Delta t/2} - e^{-\sigma\sqrt{\Delta t/2}}}{e^{\sigma\sqrt{\Delta t/2}} - e^{-\sigma\sqrt{\Delta t/2}}} \right)^2$$

$$p_d = \left(\frac{e^{\sigma\sqrt{\Delta t/2}} - e^{(r-q)\Delta t/2}}{e^{\sigma\sqrt{\Delta t/2}} - e^{-\sigma\sqrt{\Delta t/2}}} \right)^2$$

$$p_m = 1 - p_u - p_d \tag{7}$$

  where Δt is the length of time per step in the tree and is simply time to maturity divided by the number of time steps; r is the risk-free interest rate over this maturity; σ, is the corresponding volatility of the underlying; q is its corresponding dividend yield.

- Note that for p_u , p_d , and p_m to be in the interval $(0, 1)$ the following condition on Δt has to be satisfied $\Delta t < 2\frac{\sigma^2}{(r-q)^2}$.

Breakout Exercise: Create a Tree European Option Pricer

- Once the tree of prices has been calculated, the option price is found at each node largely as for the binomial model, by working backwards from the final nodes to the present node (t_0).
- The difference being that the option value at each non-final node is determined based on the three - as opposed to two - later nodes and their corresponding probabilities.
- Use the classes introduced in class and finish the SimpleTrinomialTree class to price European Option with the following parameters:
  - Expiry: 1
  - Spot: 50
  - Strike: 50
  - Risk free rate: 0.05
  - Dividend: 0.08
  - Vol: 0.3
  - NumberOfPeriods: 50

Price American Options by Random Tree Simulation

- Broadie and Glasserman (1997) proposed the method of the simulated tree to price American options, which can derive the upper and lower bounds for American options. This combination makes it possible to measure and control errors as the computational effort increases.
- The main drawback of the random tree method is that its computational requirements grow exponentially in the number of exercise dates m, so the method is applicable only when m is small.
- Nevertheless, for problems with small m it is very effective, and it also serves to illustrate a theme of managing scoures of high and low bias.

- **The Dynamic Programming Formulation:** Let $\tilde{h}_i$ denote the payoff function for exercise at t_i, which we now allow to depend on i. Let $\tilde{V}_i(x)$ denote the value of the option at t_i given $X_i = x_i$ (the option has not been exercised).
- We are ultimately interested in $\tilde{V}_0(X_0)$. This value is determined recursively as follows:

$$\tilde{V}_m(x) = \tilde{h}_m(x) \quad (8)$$

$$\tilde{V}_{i-1}(x) = max\{\tilde{h}_{i-1}(x), E[D_{i-1,i}(X_i)\tilde{V}_i(X_i)|X_{i-1} = x]\}, \quad (9)$$

  $i = 1, ..., m$ and we have introduced the notation $D_{i-1,i}(X_i)$ for the discount factor from t_{i-1} to t_i.
- It states that the option value at expiration is given by the payoff function $\tilde{h}_m$; and at time $(i-1)$th exercise date the option value is the maximum of the immediate exercise value and the expected present value of continuing.

High Estimator:

- As its name suggests, the random tree method is based on simulating a tree of paths of the underlying Markov chain $X_0, X_1, ..., X_m$.
- Fix a branching parameter $b \geq 2$. From the initial state X_0, simulate b independent successor states $X_1^1, ..., X_1^b$ all having the law of X_1.
- From each X_1^i, simulate b independent successors $X_2^{i1}, ..., X_2^{ib}$ from the conditional law of X_2 given $X_1 = X_1^i$.
- From each $X_2^{i_1 i_2}$, generate b successors $X_3^{i_1 i_2 1}, ..., X_3^{i_1 i_2 b}$, and so on.
- We denote a generic node in the tree at time step i by $X_i^{j_1 j_2 \cdots j_i}$. The superscript indicates that this node is reached by following the j_1-th branch out of X_0, the j_2th branch out the next node, and so on.

- Although it is not essential that the branching parameter remain fixed across time steps, this is a convenient simplification in discussing the method. From the random tree method we define high and low estimators at each node by backward induction.
- We use formulation (9). Thus, $\hat{h}_i$ is the discounted payoff function at the ith exercise date, and the discounted option value satisfies $\hat{V}_m \equiv \hat{h}_m$,

$$\tilde{V}_i(x) = max\{\tilde{h}_i(x), E[\tilde{V}_{i+1}(X_{i+1})|X_i = x]\}, i = 1, ..., m-1 \quad (10)$$

- Write $\hat{V}_i^{j_1 \cdots j_i}$ for the value of the high estimator at node $X_i^{j_1 \cdots j_i}$. At the terminal nodes we set

$$\hat{V}_i^{j_1 \cdots j_m} = h_m(X_m^{j_1 \cdots j_m}). \quad (11)$$

- Working backward, we then set

$$\hat{V}_i^{j_1..j_i} = max \left\{ h_i(X_i^{j_1...j_i}), \frac{1}{b} \sum_{j=1}^{b} \hat{v}_{i+1}^{j_1...j_i j} \right\}. \tag{12}$$

- Formulation (12) is based on successor nodes, so the estimator is unfairly peeking into the future in making its decision. To remove this source of bias, we need to separate the exercise decision from the value received upon continuation. A new estimator can be defined as follows: At all terminal nodes, set the estimator equal to the payoff at that node:

$$\hat{v}_m^{j_1..j_m} = h_m(X_m^{j_1...j_m}). \tag{13}$$

- At node $j_1 j_2 ... j_i$ at time step i, and for each $k = 1, ..., b$, set

$$\hat{v}_{ik}^{j_1 \cdots j_i} = h_i(X_i^{j_1 \cdots j_i}), \text{ if } \frac{1}{b}\sum_{j=1}^{b} \hat{v}_{i+1}^{j_1 j_2 \cdots j_i j} \leq h_i(X_i^{j_1 j_2 \cdots j_i});$$

$$= \hat{v}_{i+1}^{j_1 j_2 \cdots j_i k} \text{ otherwise} \qquad (14)$$

- Then set

$$\hat{v}_i^{j_1 \cdots j_i} = \frac{1}{b}\sum_{k=1}^{b} \hat{v}_{i,k}^{j_1 \cdots j_i} \qquad (15)$$

- Working backward, we then set

$$\hat{V}_i^{j_1 \cdots j_i} = max\left\{ h_i(X_i^{j_1 \cdots j_i}), \frac{1}{b}\sum_{k=1}^{b} \hat{v}_{i,k}^{j_1 \cdots j_i} \right\} \qquad (16)$$

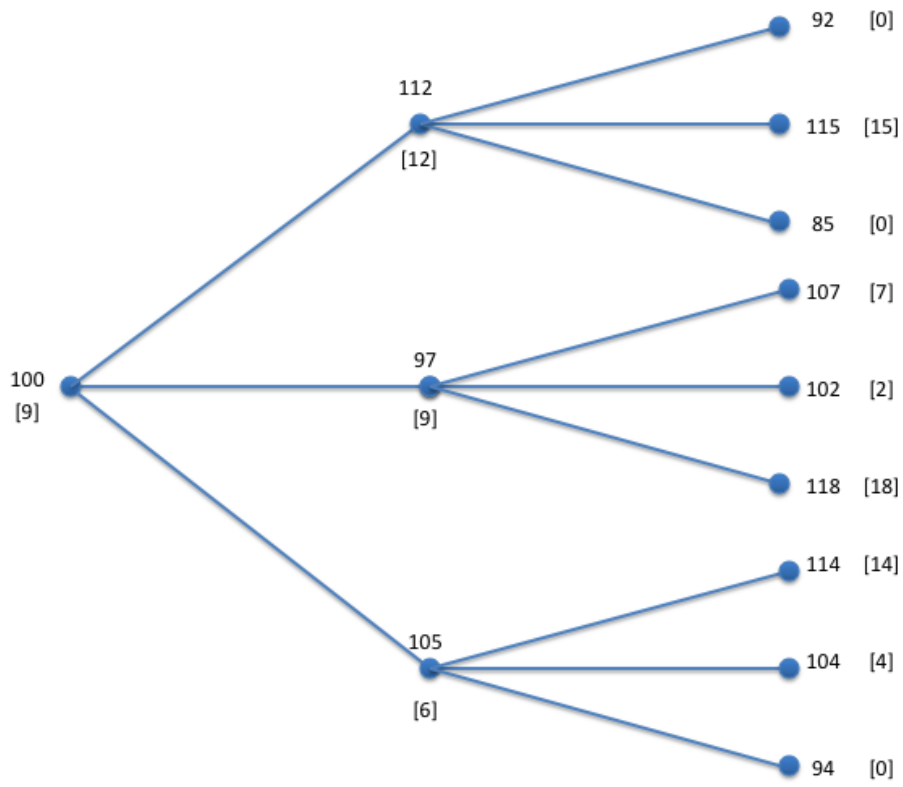

Figure: Illustration of random tree and high estimator for a call option with a single underlying asset and a strike price of 100. Labels at each node show the level of the underlying asset and (in brackets) the value of the high estimator.

- In other words, the high estimator is simply the result of applying ordinary dynamic programming to the random tree, assigning equal weight to each branch. Its calculation is illustrated in Figure (2) with $h_i(x) = (x - 100)^+$.
- A simple induction argument demonstrates that the high estimator is indeed biased high at every node, in the sense that

$$\mathbf{E}[\hat{V}_i^{j_1 \cdot j_i} | X_i^{j_1 \cdot j_i}] = \mathbf{E}[V_i^{j_1 \cdot j_i}] \tag{17}$$

Low Estimator:

- The high bias of the high estimator may be attributed to its use of the same information in deciding whether to exercise as in estimating the continuation value. This is implicit in the dynamic programming recursion (16).
- The first term inside the maximum is the immediate-exercise value, the second term is the estimated continuation value, and in choosing the maximum the estimator is deciding whether to exercise or continue. But the estimated continuation value is based on successor nodes, so the estimator is unfairly peeking into the future in making its decision.
- To remove this source of bias, we need to separate the exercise decision from the value received upon continuation. This is the key to removing high bias in all Monte Carlo methods for pricing American options.

- To simplify the discussion, consider the related problem estimating

$$max(a, \mathbf{E}[\mathbf{Y}])$$

from i.i.d. replications $Y_1, ..., Y_b$, for some constant a and random varaible Y. This is a simplified version of the problem we face at each node in the tree, with a corresponding to the immediate exercise value and $\mathbf{E}[\mathbf{Y}]$ the continuation value.

- The estimator $max(a, \bar{Y})$, with $\bar{Y}$ the sample mean of the Y_i is biased high:

$$\mathbf{E}[max(a, \bar{Y})] \geq max(a, \mathbf{E}[\bar{Y}]) = max(a, \mathbf{E}[Y])$$

This corresponds to the high estimator of the previous section.

- Suppose that we instead separate the Y_i into two disjoint subsets and calculate their sample means $\bar{Y}_1$ and $\bar{Y}_2$; these are independent of each other. Now set

$$\hat{v} = \begin{cases} a & \text{if } \bar{Y}_1 \leq a; \\ \bar{Y}_2 & \text{otherwise} \end{cases}$$

  This estimator uses $\bar{Y}_1$ to decide whether to "exercise", and if it decides not to, it uses $\bar{Y}_2$ to estimate the "continuation" value.

- Its expectation is

$$\mathbf{E}[\hat{v}] = P(\bar{Y}_1 \leq a)a + (1 - P(\bar{Y}_1 \leq a))\mathbf{E}[Y] \leq max(a, \mathbf{E}[Y]), \quad (18)$$

  so the estimator is indeed biased low. If $a \neq \mathbf{E}[Y]$, then $P(\bar{Y}_1 \to \mathbf{1}\{\mathbf{E}[Y] < a\}$ and $\mathbf{E}[\bar{v}] \to max(a, \mathbf{E}[Y])$ as the number of replications used to calculate $\bar{Y}_1$ increases.

- If the number of replications used to calculate $\bar{Y}_2$ also increases then $\hat{v} \rightarrow max(a, \mathbf{E}[Y])$. Thus, in this simplified setting we can easily produce a consistent estimator that is biased low.
- Broadie and Glasserman (1997) use a slightly different estimator. They use all but one of the Y_i to calculate $\bar{Y}_1$ and use the remaining ones for $\bar{Y}_2$; they then average the result over all b ways of leaving out one of the Y_i.
- The estimator is defined as follows. At all terminal nodes, set the estimator equal to the payoff at that node:

$$\hat{v}_m^{j_1 j_2 \cdots j_m} = h_m(X_m^{j_1 j_2 \cdots j_m})$$

At node $j_1 j_2 \ldots j_i$ at time step i, and for each $k = 1, \ldots, b$, set

$$\hat{v}_{ik}^{j_1 j_2 \cdots j_i} = \begin{cases} h_i(X_i^{j_1 j_2 \cdots j_i}) & \text{if} \frac{1}{b-1} \sum_{j=1; j \neq k}^{b} \hat{v}_{i+1}^{j_1 j_2 \cdots j_i j} \leq h_i(X_i^{j_1 j_2 \cdots j_i}); \\ \hat{v}_{i+1}^{j_1 j_2 \cdots j_i k} & \text{otherwise} \end{cases} \quad (19$$

- We then set

$$\hat{v}_i^{j_1 j_2 \cdots j_i} = \frac{1}{b} \sum_{k=1}^{b} \hat{v}_{ik}^{j_1 j_2 \cdots j_i}. \tag{20}$$

  The estimator of the option price at the current time and state is $\hat{v}_0$.

- The calculation of the low estimator is illustrated in Figure (3). Consider the third node at the first exercise date. When we leave out the first successor we estimate a continuation value of $(4+0)/2 = 2$ so we exercise and get 5. If we leave out the second successor node we continue (because $7 > 5$) and get 4.

- In the third case we continue and get 0. Averaging the three payoffs $5, 4,$ and 0 yields a low estimate of 3 at that node.

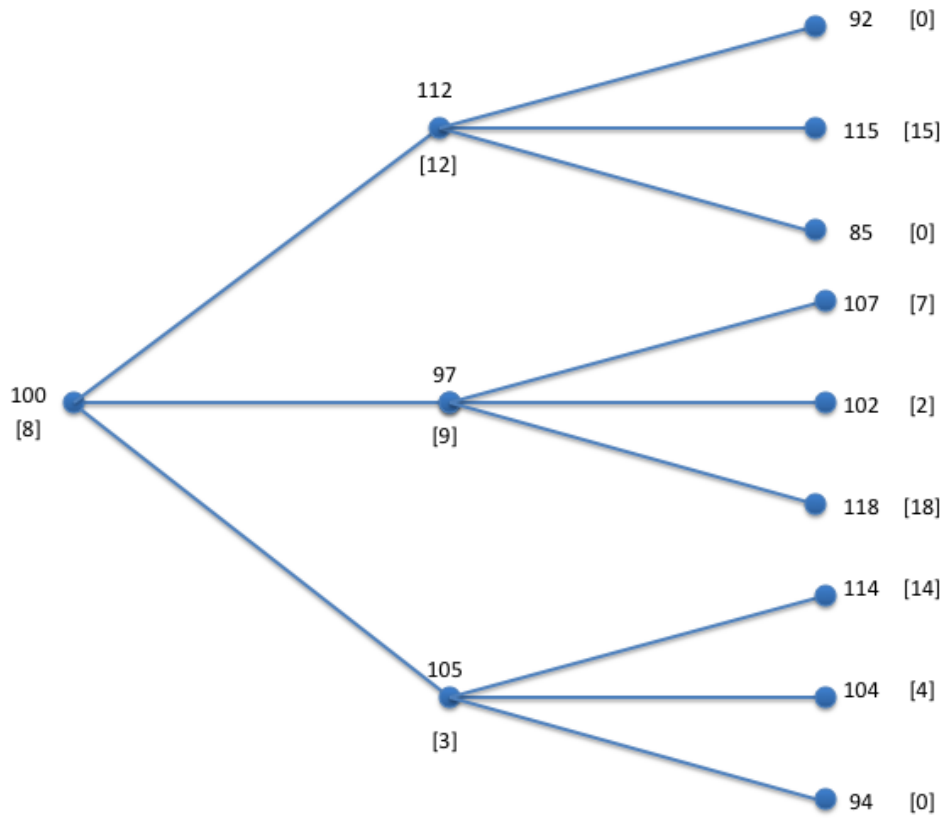

Figure: Labels at each node show the level of the underlying asset and the value of the low estimator in brackets.

- An induction argument similar to the one used for the high estimator and using the observation in (18) verifies that $\hat{v}_0$ is indeed biased low.
- From n independent replications of the random tree we can calculate the sample mean $\bar{v}_0(n, b)$ and sample standard deviation $s_v(n, b)$ of n independent replications of the low estimator. We can then form a $1 - \delta$ confidence interval for $\mathbf{E}[\hat{v}_0]$,

$$\bar{v}_0(n, b) \pm z_{\delta/2} \frac{s_v(n, b)}{\sqrt{n}}.$$

- Taking the lower limit for the low estimator and the upper limit for the high estimator, we get the interval

$$\left(\bar{v}_0(n, b) \pm z_{\delta/2} \frac{s_v(n, b)}{\sqrt{n}}, \bar{V}_0(n, b) \pm z_{\delta/2} \frac{s_V(n, b)}{\sqrt{n}} \right).$$

Random Tree Method Implementation

- A naive implementation of the random tree method generates all m^b nodes (over m steps with branching parameter b) and then computes high and low estimators recursively as described before. By noting that the high and low values at each node depend only on the subtree rooted at that node, we can dramatically reduce the storage requirements of the method. It is never necessary to store more than $mb + 1$ nodes at a time.
- **Depth-First Processing**: Recall that we may label nodes in the tree through a string of indices $j_1 j_2 ... j_i$ each taking values in the set $\{1, ..., b\}$. The string $j_1 j_2 ... j_i$ labels the node reached by following the j_1th branch out of the root node, then the j_2th branch out of the node reached at step 1, and so on.

Random Tree Method Implementation

- In the depth-first algorithm, we follow a single branch at a time rather than generating all branches simultaneously.
- Consider the case of a four-step tree. We begin by generating the following nodes:

$$1, 11, 111, 1111.$$

- At this point we have reached the terminal step and can go no deeper, so we generate nodes

$$1112, ..., 111b.$$

- From these values, we can calculate high and low estimators at node 111. We may now discard all b successors of node 111. Next we generate 112 and its successors.

Random Tree Method Implementation

- Next we generate 112 and its successors.

$$1121, 1122, ..., 112b.$$

- We discard these after using them to calculate high and low estimators at 112.
- We repeat the process to calculate the estimators at nodes $113, ..., 11b$. These in turn can be discarded after we use them to calculate high and low estimators at node 11.
- We repeat the process to compute estimators at nodes $12, .., 1b$ to get estimators at node 1, and then to get estimators at nodes $2, ..., b$ and finally at the root node.

Random Tree Method Pruning and Variance Reduction

- Broadie et al. investigate potential enhancements of the random tree method, including the use of variance reduction techniques in combination with a pruning technique for reducing the computational burden of the method.
- Their pruning technique is based on the observation that branching is needed only where the optimal exercise decision is unknown. If we know it is optimal not to exercise at a node, then it would suffice to generate a single branch out that node.
- When we work backward through the tree, the value we assign to that node for both the high and low estimators is simply the (discounted) value of the corresponding estimator at the unique successor node.
- By pruning branches, we reduce the time needed to calculate estimators from a tree.

Random Tree Method Pruning and Variance Reduction

- By pruning branches, we reduce the time needed to calculate estimators from a tree. But how can we determine that the optimal decision at a node is to continue?
- Broadie et al. suggest the use of bounds. Suppose we have the payoff function $h_i, i = 1, ..., m$, are nonnegative. Then at any node at which the payoff from immediate exercise is 0, it is optimal to continue. This simple rule is often applicable at a large number of nodes.
- Broadie et al. also discuss the use of antithetic variates and Latin hypercude sampling in generating branches. In calculating the low estimator with antithetic, they apply the steps in (19) - (20) to averages over antithetic pairs.

* Broadie et al. (1997), "Enhanced Monte Carlo estimates of American option prices", *Journal of Derivatives* 4(Fall4), 25-44.

Key Points

- The random tree pricing is based on dynamic programming principle.
- The random tree method is effective for the small number of exercising periods.
- We can re-use the pay-off class when defining products on trees.
- Pruning method can be used to improve the efficiency of the pricing engine.
- European options can be used as controls for American options.