Trinomial Tree Construction

A trinomial tree based method is presented for pricing exotic options. The model is based on a combination of techniques. that is, a tree generation technique and an appropriate backward induction pricing technique.

Since the volatility parameter in the SDE is of a piecewise constant form, the tree generation techniques may, in some cases, construct trees that are non- recombining. In the worst case, then, the space complexity of the tree generation techniques is proportional to the exponential of the number of time slices in the tree.

Let $0 = t_0 < \cdots < t_N = T$ be a partition of the time interval $[0, T]$. Furthermore suppose that the underlying security follows piecewise geometric Brownian motion, in the sense described below, over the interval $[0, T]$. Specifically, assume that the underlying security can be modeled as a process, $\left\{ S(t) | t \in [0, T] \right\}$, which, under the risk neutral probability measure, satisfies a stochastic differential equation (SDE) of the form

$$dS(t) = \mu(t)S(t)dt + \sigma(t)S(t)dW(t), \quad t \in [0, T], \tag{1}$$

where $\left\{ W(t) | t \in [0, T] \right\}$ is standard Brownian motion. Here $\mu(t)$ and $\sigma(t)$ are deterministic functions of the piecewise constant form

$$\mu(t)=\begin{cases}\mu_1, & t\in[0,t_1),\\ \vdots & \\ \mu_N, & t\in[t_{N-1},t_N],\end{cases} \quad \text{and} \quad \sigma(t)=\begin{cases}\sigma_1, & t\in[0,t_1),\\ \vdots & \\ \sigma_N, & t\in[t_{N-1},t_N],\end{cases}.$$

Each method includes a technique for constructing, based on the SDE (1), an appropriate tree of discrete prices of the underlying security. Each such technique uses a mathematical result, described below, for ensuring that branching probabilities from each tree node are appropriate (i.e., probabilities, for each node, must be non-negative and sum to one).

Consider a tree node, ω, at a time slice, t_i, where $0 \le i < N$; furthermore, assume that the logarithm of the price of the underlying security at this node is equal to $\log Sold$. We assume that node ω branches into three nodes, at time slice t_{i+1}, with respective logarithm of the price of the underlying security of the form $(\lambda+1)\Delta\log Snew$, $(\lambda)\Delta\log Snew$, and $(\lambda-1)\Delta\log Snew$ where $\lambda\in\Re$ and $\Delta\log Snew>0$.

Here $(\lambda)\Delta\log Snew$ is the value that, among all tree nodes at time t_{i+1}, is closest to $\log Sold+\hat{\mu}_{i+1}\Delta t_{i+1}$ where $\hat{\mu}_{i+1}=\mu_{i+1}-\frac{\sigma_{i+1}^2}{2}$ and $\Delta t_{i+1}=t_{i+1}-t_i$; furthermore, $(\lambda+1)\Delta\log Snew$ and $(\lambda-1)\Delta\log Snew$ are values for the two nodes closest to the node with value $(\lambda)\Delta\log Snew$. Next we associate with node ω a discrete random variable, Y, which takes the values

$$Y=\begin{cases}(\lambda+1)\Delta\log Snew, & \text{with probability } p_u,\\ \lambda(\Delta\log Snew), & \text{with probability } p_m,\\ (\lambda-1)\Delta\log Snew, & \text{with probability } p_{d.}\end{cases}$$

We seek to determine p_u, p_m and p_d, above, so that the mean and variance of the discrete random variable Y match those of the continuous random variable $\log Sold + \hat{\mu}_{i+1}\Delta t_{i+1} + \sigma_{i+1} W_{\Delta_{t_{i+1}}}$ (obtained by solving the SDE (1), with initial condition $\log S(t_i) = \log Sold$, for the time interval $[t_i, t_{i+1}]$).

By matching mean and variances as described above, and by ensuring that the probabilities sum to one, we obtain the following system of linear equations

$$\begin{cases} p_u(\lambda+1)\Delta\log Snew + p_m\lambda(\Delta\log Snew) + p_d(\lambda-1)\Delta\log Snew = \log Sold \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \hat{\mu}_{i+1}\Delta t_{i+1}, \\ p_u(\lambda+1)^2(\Delta\log Snew)^2 + p_m(\lambda)^2(\Delta\log Snew)^2 + \\ \qquad\qquad p_d(\lambda-1)^2(\Delta\log Snew)^2 = \sigma_{i+1}{}^2\Delta t_{i+1} + (\log Sold + \hat{\mu}_{i+1}\Delta t_{i+1})^2, \\ p_u + p_m + p_d = 1, \end{cases} \tag{2}$$

for the unknowns p_u, p_d and p_m. By algebraic manipulation, the linear system of equations above is equivalent to

$$\begin{cases} p_u - p_d = B - \lambda, \\ p_u(1+2\lambda) + p_d(1-2\lambda) = A^2 + B^2 - \lambda^2, \\ p_u + p_m + p_d = 1, \end{cases} \tag{3}$$

Where

$$A^2 = \frac{\sigma_{i+1}{}^2 \Delta t_{i+1}}{(\Delta \log Snew)^2} \tag{4a}$$

and

$$B = \frac{\log Sold + \hat{\mu}_{i+1} \Delta t_{i+1}}{\Delta \log Snew}. \tag{4b}$$

Notice that while the system of equations above has a unique solution, we have no guarantee that p_u, p_m and p_d will be non-negative. Next we determine a condition on A to ensure that $p_u, p_m, p_d \geq 0$.

Recall that the branching rule from node ω implies that

$$(\lambda + \frac{1}{2})\Delta \log Snew \geq \log Sold + \hat{\mu}_{i+1} \Delta t_{i+1} \geq (\lambda - \frac{1}{2})\Delta \log Snew. \tag{5}$$

Dividing (5) by $\Delta \log Snew$ and substituting B for the right hand side of (4b), we have $\lambda + \frac{1}{2} \geq B \geq \lambda - \frac{1}{2}$, which we can rewrite as

$$B = \lambda - \frac{1}{2} + x \tag{6}$$

for some $x \in [0,1]$. Solving (3) and substituting the right hand side of (4b) for B, we obtain

$$\begin{cases} p_u = \dfrac{A^2}{2} + \dfrac{x^2}{2} - \dfrac{1}{8}, \\ p_m = \dfrac{3}{4} + x - x^2 - A^2, \\ p_d = \dfrac{A^2}{2} + \dfrac{x^2}{2} - x + \dfrac{3}{8}, \end{cases} \tag{7}$$

where $x \in [0,1]$. Notice that the right hand side of (7) has no dependency on λ.

Analyzing the right hand side of (7) over the range $x \in [0,1]$, we obtain the condition

$$\frac{3}{4} \geq A^2 \geq \frac{1}{4} \tag{8}$$

on A, which ensures that $p_u, p_m, p_d \geq 0$. Notice that (8) yields an equivalent condition on $\Delta \log Snew$, that is,

$$\frac{2\sigma_{i+1}\sqrt{\Delta t_{i+1}}}{\sqrt{3}} \leq \Delta \log Snew \leq 2\sigma_{i+1}\sqrt{\Delta t_{i+1}} \ . \tag{9}$$

To summarize, for an arbitrary tree node on an arbitrary time slice, appropriate branching probabilities are given as the solution of (2) provided that condition (9) holds. In Appendix B we examine the sensitivity of solutions to (2) with respect to perturbations in the values of $\Delta \log Snew$ and λ.

In this section we present the techniques for generating a tree appropriate for pricing the barrier options described in Section 2. We consider single barrier options first.

Suppose that we have constructed a tree up to time t_i, where $1 \le i < N$. To expand the tree to the next time slice, we first define, at time t_{i+1}, an appropriate partition for the logarithm of the underlying security; then, using this partition, we determine the children and associated probabilities of all nodes at time t_i.

Note that, by an appropriate partition for the logarithm of the underlying security at time t_{i+1}, we mean a partition such that the inter-node spacing is equal to $\Delta \log Snew$ where $\Delta \log Snew$ is chosen (as in Section 3.1.1) so that branching probabilities are non-negative. Next we describe how to construct such a partition. Then we discuss how to determine the branching and corresponding probabilities for nodes at time t_i.

To define a partition at time t_{i+1} (with uniform spacing, $\Delta \log Snew$, which satisfies the inequality (9)), we first determine whether, for some nodes at time t_i, there is a branch that crosses the barrier at time t_{i+1}. This determination is made by checking certain conditions, defined in Appendix A, based on the branching rule (for nodes on the old time slice to nodes on the new time slice) defined. I f we determine that the barrier will be crossed at time t_{i+1}, we generate a partition by placing a node on the actual barrier (i.e., either $H_u(t_{i+1})$ or $H_d(t_{i+1})$) and all other nodes offset from the barrier by integer multiples of $\Delta \log Snew$. Otherwise, if the barrier will not be crossed, we define an artificial barrier at time t_{i+1} (see Appendix A); we then generate a partition by placing a node exactly on the artificial barrier and all other nodes offset by integer multiples of $\Delta \log Snew$ from this barrier (here the artificial barrier simply acts as a point of reference for generating the partition). We use for $\Delta \log Snew$ a value close to the upper bound, $2\sigma_{i+1}\sqrt{\Delta t_{i+1}}$, in the inequality (9).

Once an appropriate partition has been defined at time t_{i+1}, we then determine, according to the branching rule presented in Section 3.1.1, the children of each node at time t_i. Suppose that a particular node at time t_i (with value for the logarithm of the underlying security equal to $\log Sold$) branches to nodes at time t_{i+1} with values for the logarithm of the underlying security equal to a_u, a_m and a_d, respectively, where $a_u > a_m > a_d$. Then, from (2), appropriate branching probabilities are determined by solving the system of linear equations

$$\begin{cases} p_u a_u + p_m a_m + p_d a_d = \log Sold + \hat{\mu}_{i+1}\Delta t_{i+1}, \\ p_u {a_u}^2 + p_m a_m^2 + p_d a_d^2 = {\sigma_{i+1}}^2 \Delta t_{i+1} + (\log Sold + \hat{\mu}_{i+1}\Delta t_{i+1})^2, \\ p_u + p_m + p_d = 1, \end{cases}$$

for the unknowns p_u, p_d and p_m. The numerical conditioning of the system of linear equations above should be checked, however, to ensure the accuracy of the computed solution.

The tree construction technique for double barrier options is based on a similar approach as for single barrier options. That is, if the tree has been constructed up to time t_i, an appropriate partition for the underlying security is defined at time t_{i+1}; then branches and associated probabilities are determined for nodes on the old time slice. We describe these techniques next.

Suppose that the tree has been generated up to time t_i, where $1 \le i < N$. If neither barrier is crossed at time t_{i+1}, then an artificial barrier is defined, at time t_{i+1}, and used (as a point of reference) to generate an appropriate partition. Otherwise, a partition is defined that places nodes on both barriers (see below). Once a partition is defined, branching and corresponding probabilities are determined as above.

Next we show how to construct a uniformly spaced partition for the logarithm of the underlying security price at a time slice, t_i, so that nodes are placed on both the upper and lower barriers. We begin by placing a node on the logarithm of the upper barrier, $\log H_u(t_i)$. Then we attempt to place a node on the logarithm of the lower barrier, $\log H_d(t_i)$, making sure that condition (9) in Section 3.1.1 holds.

Let $\Delta \log Snew_{\min} = \dfrac{2\sigma_i \sqrt{\Delta t_i}}{\sqrt{3}}$ and $\Delta \log Snew_{\max} = 2\sigma_i \sqrt{\Delta t_i}$ denote, respectively, the lower and upper bounds in the inequality (9) (with respect to time slice t_i). Also let $\Delta \log H = \log H_u(t_i) - \log H_d(t_i)$ be the difference in the logarithm of the upper and lower barrier values at time t_i. Now let $j = \left\lfloor \dfrac{|\Delta \log H|}{\Delta \log Snew_{\min}} \right\rfloor$ be the number of entire $\Delta \log Snew_{\min}$ intervals contained in $|\Delta \log H|$. We seek

$$\Delta \log Snew \in [\Delta \log Snew_{\min}, \Delta \log Snew_{\max}]$$

such that

$$j(\Delta \log Snew) = |\Delta \log H|. \qquad (10)$$

Observe that the only way (10) can fail to hold is if $j(\Delta \log Snew_{\max}) < |\Delta \log H|$, that is,

$$j\sqrt{3} < \frac{|\Delta \log H|}{\Delta \log Snew_{\min}}. \tag{11}$$

Next notice that (11) is of the form $\lfloor \alpha \rfloor \sqrt{3} < \alpha$, for $\alpha \in R$, which does not hold for any $\alpha > 2$ (i.e., in the case there are at least three nodes between the upper and lower barriers).

To summarize, as long as $\left\lfloor \frac{|\Delta \log H|}{\Delta \log Snew_{\min}} \right\rfloor > 2$, we can find $\Delta \log Snew \in [\Delta \log Snew_{\min}, \Delta \log Snew_{\max}]$ such that (10) holds (i.e., we fit both barriers). In particular, we can proceed by computing $j = \left\lfloor \frac{|\Delta \log H|}{\Delta \log Snew_{\min}} \right\rfloor$. We then check that $j > 2$; if not, we then decrease $\Delta \log Snew_{\min}$ by decreasing Δt_i. Next we take $\Delta \log Snew = \frac{|\Delta \log H|}{j}$.

Reference:

https://finpricing.com/lib/IrCurveIntroduction.html