American Journal of Computational Mathematics, **2013, 3, 291-296**
Published Online December 2013 (http://www.scirp.org/journal/ajcm)
http://dx.doi.org/10.4236/ajcm.2013.34038

A New Recombination Tree Algorithm for Mean-Reverting Interest-Rate Dynamics

Peter C. L. Lin
Department of Mathematical Sciences & Financial Engineering Program,
Stevens Institute of Technology, Hoboken, USA
Email: peter.lin@stevens.edu

Received June 13, 2013; revised August 15, 2013; accepted September 12, 2013

ABSTRACT

In light of the fact that no existing tree algorithms can guarantee the recombination property for general Ornstein-Uhlenbeck processes with time-dependent parameters, a new trinomial recombination-tree algorithm is designed in this research. The proposed algorithm enhances the existing mechanisms in interest-rate modelings with the comparisons to [1,2] methodologies, and the proposed framework provides a more efficient way in discrete-time mean-reverting simulations.

Keywords: Natural Asset; Financial Value; Neural Network

1. Introduction

A general Ornstein-Uhlenbeck process is defined such that

$$\mathrm{d}G(t) = k(t)\big(a(t) - G(t)\big)\mathrm{d}t + b(t)\mathrm{d}\tilde{W}(t)$$

where $W(t)$ is a standard Brownian motion, and $k(t)$, $a(t)$, $b(t)$ are time-dependent deterministic parameters. The parameter $a(t)$ is the mean-reversion term indicating the long-term mean-reverting level with a rate $k(t)$ at time *t*. The source of randomness is described by the Brownian motion $W(t)$ multiplied by a volatility term $b(t)$.

A tree is an acyclic structure where each node has zero to multiple descendant nodes and one parent node. A recombination tree is a special tree structure of which the size grows linearly. Therefore the investing decisions, if computed recursively, have time complexity[1] at most $O(n^2)$, which is much more efficient than a general simulation method which may cost exponential amount of time. For example, a recombination tree can help us to efficiently determine the price and the buy/sell timing for an American style option by comparing the derivatives value at each tree node with its children nodes (see [4] for more details). However, designing a recombination tree algorithm for modeling interest rates is far from trivial. Here are two examples:

[1]For the terminology of computational complexity please see [3].

In [1], Hull and White provided a heuristic two-stage method for constructing an interest rate tree based on the extended-Vasicek short-rate model.[2] In the first stage the algorithm builds the framework of the tree, and in the second stage the algorithm calibrates the tree to the current interest-rate term structure. The algorithm is designed for a short-rate model; hence the tree cannot be adjusted or updated according to the markets. Also, their method cannot deal with stochastic mean-reverting parameters, and there is no guarantee that the tree is a recombination tree especially when the volatility term of the short-rate process is a decreasing function. Therefore, Hull-White's algorithm is not a good candidate.

In [2], Black, Derman, and Toy (BDT hereafter) also provided a recombination tree algorithm for short rates. Their tree is constructed recursively and calibrated to zero-coupon bond volatilities and current interest rate structure. Though the BDT tree guarantees a recombination structure, the tree is not designed for a general Ornstein-Uhlenbeck process. Therefore, the BDT tree is not a good candidate either.

In light of the fact that no existing tree algorithms can guarantee the recombination property for general Orn-

[2]For more definitions of short-rate models, see [5]. Yet, for this article we should focus only on the mathematical modeling for general Ornstein-Uhlenbeck Processes.

stein-Uhlenbeck processes, we propose a new recombination trinominal tree algorithm. The idea is to modify a standard trinominal tree (**Exhibit 1(a)**) by adding extra branches at each node. First, we denote the tree structure in black color the *center path*. Then, given a node V directly above (below) the center path, define $\langle V \rangle$ the set of nodes containing V and all the nodes down to (up to) the center path. Then we modify the tree according to the following rules: 1) the center path remains unchanged; 2) given a node V above the center path, we connect the node to all the descendant (children) nodes stemming from $\langle V \rangle$; 3) given a node V below the center path, we connect the node to all the descendant nodes stemming from all the nodes from $\langle V \rangle$. The modified tree structure is shown in **Exhibit 1(b)**. We will use the names, *spanning nodes and spanning branches*, to identify those nodes not on the center path, and branches not emanating from a center node.

The crucial key of modifying a standard trinominal tree is that we can further simplify and still keep the tree structure by adding *sibling branches*. Sibling branches are one-way streets through which we can only move up or down at a given time epoch, but not in both directions. A spanning node above the center path can reach the center path and all the nodes in between only by moving down through the sibling branches; a spanning node below the center path can reach the center path and all the nodes in between only by moving up through the sibling branches. As a result, by adding the sibling branches, each node can reach all but one descendent nodes via its sibling branches. So, in the simplified tree structure, each spanning node will have only one time transition descendant branch. The final tree structure is shown in **Exhibit 1(c)**. The algorithm is given below. The proof of the correctness of the algorithm for simulation is given in Section 3.

2. Algorithm

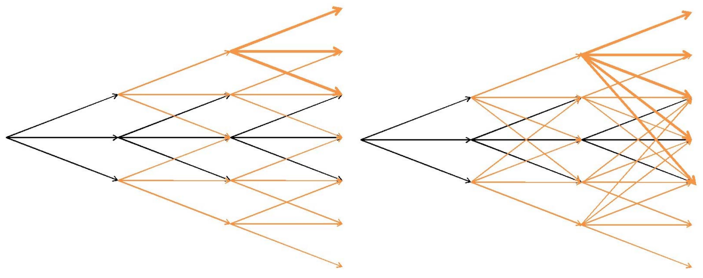
Now we give a full description of the algorithm. Let $g_{j,j}$ denote the node on the center path at time t_j, and let $g\left(t_j, j\right)$ denote the value of node $g_{j,j}$. Therefore, if $g:\mathbb{R}\times\mathbb{Z}\mapsto\mathbb{R}$ is represented as a function, then it indicates the value of the node. Let $g_{j,j+k}$ and $g\left(t_j, j+k\right)$ denote the k-th node above center node $g_{j,j}$ and the value of $g_{j,j+k}$ respectively. Similarly, let $g_{j,j-k}$ and $g\left(t_j, j-k\right)$ denote the k-th node below center node $g_{j,j}$ and the value of $g_{j,j-k}$ respectively. Moreover, if we use capital letter $G\left(t_j,\omega\right)$, then it represents a random variable of the tree value at time t_j. To shorthand the notation, the expectation value $G\left(t_{j+1}\right)$ conditional on the position of $G\left(t_j\right)$ is written as

$$\mathrm{E}\left[G\left(t_{j+1}\right)\middle|G\left(t_j\right)\right]. \tag{1}$$

Define the conditional expectation at node $g_{j,j}$ to be

$$M\left(t_j, j\right)=E\left[G\left(t_{j+1}\right)\middle|G\left(t_j\right)=g_{j,j}\right] \tag{2}$$

Since the volatility term in stochastic-splines model is assumed to be a deterministic function, the conditional variance is the same for all nodes at a given time, *i.e.* at time t_j, the conditional variance


(a) (b)

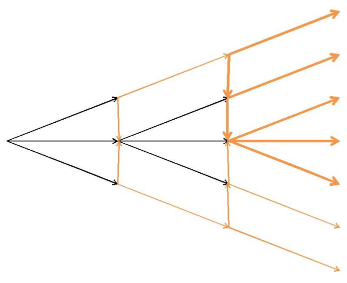

(c)

Exhibit 1. (a) original recombination trinominal tree; (b) modified tree; (c) simplified Tree.

Algorithm 1 Recombination Tree

Require: MAXLEVEL (Tree Size)

```
1:  {Stage One – Center Path}
2:  For j = 1 to MAXLEVEL do
3:      $\Delta x_j \leftarrow \sqrt{V_j}$
4:      $h \leftarrow round\left(\frac{M(t_j, j)}{\Delta x_j}\right)$
5:      Set $\begin{cases} g(t_j, j) \leftarrow h \times \Delta x_j \\ g(t_j, j+1) \leftarrow (h+1) \times \Delta x_j \\ g(t_j, j-1) \leftarrow (h-1) \times \Delta x_j \end{cases}$.
6:      Set $\begin{cases} p_u \leftarrow \frac{1}{6} + \frac{(M(t_j, j) - g(t_j, j))^2}{6V_{t_j}^2} + \frac{M(t_j, j) - g(t_j, j)}{2\sqrt{3}V_{t_j}} \\ p_n \leftarrow \frac{2}{3} - \frac{(M(t_j, j) - g(t_j, j))^2}{3V_{t_j}^2} \\ p_d \leftarrow \frac{1}{6} + \frac{(M(t_j, j) - g(t_j, j))^2}{6V_{t_j}^2} - \frac{M(t_j, j) - g(t_j, j)}{2\sqrt{3}V_{t_j}} \end{cases}$.
7:  end for
8:  {Stage Two – Spanning Branches}
9:  for i = 1 to MAXLEVEL do
10:     for j = 1 to $2 \times j - 1$ do
11:         {Move to the j-th vertex below the center path}
12:         Find $x \in \left[M_{i,j-1} - \sqrt{V_i}, M_{i,j-1} + \sqrt{V_i}\right]$ where
            $\frac{x - M_{i,j-1}}{x - M_{i,j}} = \frac{V_i - (x - M_{i,j-1})^2}{V_j + (M_{i,j} - M_{i,j-1})^2 - (x - M_{i,j-1})^2}$
13:         $g(t_{i+1}, j-2) \leftarrow x$
14:         $p(g_{i,j-1}, g_{i,j}) \leftarrow \frac{x - M_{i,j-1}}{x - M_{i,j}}$
15:         $p(g_{i,j-1}, g_{i+1,j-2}) \leftarrow 1 - \frac{x - M_{i,j-1}}{x - M_{i,j}}$
16:         {Move to the j-th vertex above the center path}
17:         Find $x \in \left[M_{i,j+1} - \sqrt{V_i}, M_{i,j+1} + \sqrt{V_i}\right]$ where
            $\frac{x - M_{i,j+1}}{x - M_{i,j}} = \frac{V_i - (x - M_{i,j+1})^2}{V_j + (M_{i,j} - M_{i,j+1})^2 - (x - M_{i,j+1})^2}$
```

Continued

```
18:                  $g(t_{i+1}, j+2) \leftarrow x$
19:                  $p(g_{i,j+1}, g_{i,j}) \leftarrow \frac{x - M_{i,j+1}}{x - M_{i,j}}$
20:                  $p(g_{i,j+1}, g_{i+1,j+2}) \leftarrow 1 - \frac{x - M_{i,j+1}}{x - M_{i,j}}$
21:          end for
22:      end for
```

$$V_j = \mathrm{Var}\left(G\left(t_{j+1}\right)\middle|G\left(t_j\right)\right). \tag{3}$$

The idea of the recombination algorithm is to construct the center path first including the node values and branch probabilities, then determine the values of the spanning nodes, the probabilities on the spanning branches, and the probabilities on the sibling branches. The details are given in Algorithm 1. However, the algorithmshows that each tree is designed for simulating one coefficient process; if we have *N* coefficient, we will need to build *N* trees altogether if coefficient processes are correlated. After constructing the coefficient recombination "forest", we can simulate the interest rate curve efficiently.

The justification of the algorithm is given in the next Section.

3. Verification

First we look at the *first part* of the algorithm and some notations. The first stage of the algorithm follows the standard Hull-White methodology (see [1]) and provides the backbone of the tree. Let $g_{j,j}$ denote the node on the center path at time t_j, and let $g(t_j, j)$ denote the value of node $g_{j,j}$. Therefore, if g is represented as a function, then it indicate the value of the node. Let $g_{j,j+k}$ and $g(t_j, j+k)$ denote the *k*-th node above center node $g_{j,j}$ and the value of $g_{j,j+k}$ respectively. Similarly, let $g_{j,j-k}$ and $g(t_j, j-k)$ denote the *k*-th node below center node $g_{j,j}$ and the value of $g_{j,j-k}$ respectively. Moreover, if we use capital letter $G(t_j)$, then it represents a random variable of the tree value at time t_j. To shorthand the notation, the expected value $G(t_{j+1})$ conditional on the position of $G(t_j)$ is written as

$$\mathrm{E}\left[G\left(t_{j+1}\right)\middle|G\left(t_j\right)\right]. \tag{4}$$

Now we move to the *second part* of the algorithm. The tree branches besides the central path are called spanning branches and spanning nodes. The second stage of the algorithm adopts the ideas of the law of total expectations and the law of total variances to assign the values and probabilities of spanning nodes and branches. The procedure is done recursively. Therefore we just need to look at the cases when $k = -1$ and $k = 1$. Given a node $g_{j+1,j+2}$ spanning from node $g_{j,j+1}$ at time t_j, we denote the conditional expectation and conditional variance at node $g_{j,j+1}$ to be $M_{j,j+1}$ and $V_{j,j+1}$ respectively. Denote the probability p to be the probability moving down from $g_{j,j+1}$ to $g_{j,j}$ and $(1-p)$ to be the probability moving through the spanning branch from $g_{j,j+1}$ to $g_{j+1,j+2}$.

We can recall the law of total expectation which states

$$\mathrm{E}[X] = \sum p(Y = y)\,\mathrm{E}\left[X\middle|Y = y\right]. \tag{5}$$

If we let

$$X = \tilde{\mathrm{E}}\left[G\left(t_{j+1}\right)\middle|G\left(t_j\right) = g_{j,j+1}\right], \tag{6}$$

and *Y* denotes the random variable such that

$$Y = \begin{cases} 0, & \text{if moving to the spanning brance} \\ 1, & \text{otherwise} \end{cases}, \tag{7}$$

then

$$\begin{aligned} M_{j,j+1} &= \tilde{\mathrm{E}}\left[G\left(t_{j+1}\right)\middle|G\left(t_j\right) = g_{j,j+1}\right] \\ &= p\tilde{\mathrm{E}}\left[G\left(t_j\right)\middle|G\left(t_j\right) = g_{j,j}\right] \\ &\quad + (1-p)\tilde{\mathrm{E}}\left[G\left(t_{j+1}, j+2\right)\middle|G\left(t_j\right) = g_{j,j+1}\right] \\ &= pM_{j,j} + (1-p)\,g\left(t_{j+1}, j+2\right), \end{aligned} \tag{8}$$

which shows

$$p = \frac{g\left(t_{j+1}, j+2\right) - M_{j,j+1}}{g\left(t_{j+1}, j+2\right) - M_{j,j}}. \tag{9}$$

The task of deriving the relationship between $g(j+1, j+2)$ and p from the law of total variance is more complicated. We will show the the result first, then break into each part. Recall the law of total variance which states

$$\operatorname{Var}(X) = \mathrm{E}\left[\operatorname{Var}(X|Y)\right] + \operatorname{Var}\left(\mathrm{E}\left[X|Y\right]\right). \quad (10)$$

Similarly we let

$$X = \tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j)\right], \quad (11)$$

and Y denotes the random variable such that

$$Y = \begin{cases} 0, & \text{if moving to spanning brance} \\ 1, & \text{otherwise} \end{cases}. \quad (12)$$

If the following statement is true:

$$\begin{aligned}
&\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right]\right) \\
&= \tilde{\mathrm{E}}\left[\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y\right)\right] \\
&\quad + \widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y\right]\right) \quad (13) \\
&= pV_j + p\left(M_{j,j} - M_{j,j+1}\right)^2 \\
&\quad + (1-p)\left(g(j, j+1) - M_{j,j+1}\right)^2,
\end{aligned}$$

then we have

$$p = \frac{V_j - \left(g(j, j+1) - M_{j,j+1}\right)^2}{V_j + \left(M_{j,j} - M_{j,j+1}\right)^2 - \left(g(j, j+1) - M_{j,j+1}\right)^2}. \quad (14)$$

Examining the first term, pV_j, on the right-hand-side of Equation (13),

$$\begin{aligned}
&\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 0\right) \\
&= \widetilde{\operatorname{Var}}\left(g(j+1, j+2)\right) = 0
\end{aligned} \quad (15)$$

since there is only one choice moving from $g_{j,j+1}$ to $g_{j+1,j+2}$. On the other hand,

$$\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 1\right) = V_j, \quad (16)$$

and we know this value recursively. So

$$\begin{aligned}
&\tilde{\mathrm{E}}\left[\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y\right)\right] \\
&= (1-p)\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 0\right) \\
&\quad + p\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 1\right) \\
&= p\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 1\right) = pV_j.
\end{aligned} \quad (17)$$

Next, the second and third terms. Since

$$\begin{aligned}
&\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = G(t_j, j+1)\right] \middle| Y\right]\right] \\
&= \tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = G(t_j, j+1)\right] = M_{j,j+1},
\end{aligned} \quad (18)$$

we have

$$\begin{aligned}
&\widetilde{\operatorname{Var}}\left(\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y\right]\right) \\
&= p\left(\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 1\right] - M_{j,j+1}\right)^2 \\
&+ (1-p)\left(\tilde{\mathrm{E}}\left[\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] \middle| Y = 0\right] - M_{j,j+1}\right)^2 \\
&= p\left(\tilde{\mathrm{E}}\left[G(t_{j+1}) \middle| G(t_j) = g_{j,j+1}\right] - M_{j,j+1}\right)^2 \\
&+ (1-p)\left(g(j+1, j+2) - M_{j,j+1}\right)^2 \\
&= p\left(M_{j,j} - M_{j,j+1}\right)^2 + (1-p)\left(g(j+1, j+2) - M_{j,j+1}\right)^2.
\end{aligned} \quad (19)$$

Now we have two equations and two unknown p and $g(t_{j+1}, j+2)$ in the following

$$\begin{cases} p = \dfrac{g(t_{j+1}, j+2) - M_{j,j+1}}{g(t_{j+1}, j+2) - M_{j,j}} \\[2ex] q = \dfrac{V_j - \left(g(j+1, j+2) - M_{j,j+1}\right)^2}{V_j + \left(M_{j,j} - M_{j,j+1}\right)^2 - \left(g(j+1, j+2) - M_{j,j+1}\right)^2} \end{cases}. \quad (20)$$

However, p must be a number between 0 and 1. And we now show that the equations indeed yield a solution such that $p \in [0,1]$. First, the case where $M_{j,j+1} \ge M_{j,j}$ and write

$$f_1(x) = \frac{x - M_{j,j+1}}{x - M_{j,j}} \quad (21)$$

and

$$f_2(x) = \frac{V_j - \left(x - M_{j,j+1}\right)^2}{V_j + \left(M_{j,j} - M_{j,j+1}\right)^2 - \left(x - M_{j,j+1}\right)^2}. \quad (22)$$

Since $M_{j,j+1} \ge M_{j,j}$, for any $x \ge M_{j,j+1}$

$$0 \le \frac{x - M_{j,j+1}}{x - M_{j,j}} \le 1 \quad (23)$$

and $f_1(x)$ is continuous and monotonically increasing. On the other hand, for any $x \in \left[M_{j,j+1}, M_{j,j+1} + \sqrt{V_j}\right]$,

$$0 \le \frac{V_j - \left(x - M_{j,j+1}\right)^2}{V_j + \left(M_{j,j} - M_{j,j+1}\right)^2 - \left(x - M_{j,j+1}\right)^2} \le 1 \quad (24)$$

and $f_2(x)$ is a continuous and monotonically decreasing function. Since

$$f_1\left(M_{j,j+1}\right) = 0, \qquad f_1\left(M_{j,j+1} + \sqrt{V_j}\right) > 0 \quad (25)$$

and

$$f_2\left(M_{j,j+1}\right) > 0, \qquad f_x\left(M_{j,j+1} + \sqrt{V_j}\right) = 0, \tag{26}$$

we know that there must exist a unique
$\hat{x} \in \left[M_{j,j+1}, M_{j,j+1} + \sqrt{V_j}\right]$ such that

$$f_1\left(\hat{x}\right) = f_x\left(\hat{x}\right) = p \in [0,1]. \tag{27}$$

Alternatively, the proof is similar for the case when $M_{j,j+1} < M_{j,j}$ except the solution exists in $\left[M_{j,j+1} - \sqrt{V_j}, M_{j,j+1}\right]$. The uniqueness and existence of the solution p and $g\left(t_{j+1}, j+2\right)$ help us solve the equations fast.

4. Conclusion

This research proposes a new trinomial recombination-tree algorithm for simulating general Ornstein-Uhlenbeck processes with time-dependent parameters. We show that there is an equivalent recombination-tree structure to simulate the mean-reverting interest-rate dynamics. Detailed algorithm and justification are given.

REFERENCES

[1] J. Hull and A. White, "Pricing Interest-Rate-Derivative Securities," *Review of Financial Studies*, Vol. 3, No. 4, 1990, pp. 573-592. http://dx.doi.org/10.1093/rfs/3.4.573

[2] F. Black, E. Derman and W. Toy, "A One-Factor Model of Interest Rates and Its Application to Treasury Bond Options," *Financial Analysts Journal*, Vol. 46, No. 1, 1990, pp. 33-39. http://dx.doi.org/10.2469/faj.v46.n1.33

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," 3rd Edition, The MIT Press, Cambridge, 2009.

[4] J. Hull, "Options, Futures, and Other Derivatives," 7th Edition, Prentice Hall, Upper Saddle River, 2008.

[5] D. Brigo and F. Mercurio, "Interest Rate Models-Theory and Practice," 2nd Edition, Springer, New York, 2006.