On Accurate Trinomial GARCH Option Pricing Algorithms

Advisor: Prof. Yuh-Dauh Lyuu

Chun-Yang Liu

Department of Computer Science and Information Engineering
National Taiwan University

Abstract

The GARCH model has been successful in describing the volatility dynamics of asset return series. However, tree-based GARCH option pricing algorithms suffer from exponential running time, inaccuracy, or other problems. Lyuu and Wu proved that the trinomial-tree option pricing algorithms of Ritchken and Trevor (1999) and Cakici and Topyan (2000) explode exponentially when the number of partitions per day, n, exceeds a threshold determined by the GARCH parameters. The improved algorithm of Lyuu and Wu (2003) still contains some problems. For example, the option prices suffer a trend to deviate from true values as n increases. This thesis proposes a new methodology to further improve the Lyuu-Wu algorithm by addressing this problem. We will confirm our algorithm's efficiency and accuracy with numerical experiments.

Keywords: GARCH model, trinomial tree, option pricing, cubic interpolation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Efficient numerical algorithms play a critical role in derivatives pricing because it is often imperative to obtain price first, particularly when prices change quickly. In both theory and practice, computational efficiency is measured by running time. The algorithms can be separated into two parts: polynomial-time algorithms and exponential-time algorithms (Papadimitriou (1995)). Because exponential function grows extremely fast, exponential-time algorithms suffer from combinatorial explosion and should be avoided wherever possible.

In the numerical pricing of derivatives, the continuous diffusion process for the asset price is turned into a tree first. Derivatives are priced on the tree by standard backward induction. The lognormal diffusion, for instance, gives rise to the well-known CRR binomial tree of Cox, Ross, and Rubinstein (1979). Two critical features of the CRR tree, as well as its many binomial and trinomial variants, stand out: It recombines and an N-period tree contains only $O(N^2)$ node. As a result, simple derivatives such as vanilla option, barrier options, and lookback options can be priced efficiently as surveyed in Lyuu (2002). However, a polynomial-sized tree may still result in an exponential-time pricing algorithm if the derivative's payoff is complex. The Asian option with a payoff depending on the arithmetic price average fits the characterization. The large number of extra states needed by the Asian option's path-dependent feature make pricing on an N-period tree take time exponential in N. Approximations are used for such derivatives to regain efficiency. Of course, approximation algorithms must not sacrifice accuracy. Important numerical techniques include the tree method with interpolation such as Hull and White (1993), the PDE method such as Forsyth, et al. (2002), and the linear-programming technique such as Dempster and Richards (2000).

A more fundamental problem will emerge when the explosion arises from

the model itself. If the model generates trees that do not recombine, pricing becomes expensive even for the simplest of derivatives. For example, when the volatility is not constant, such as the interest rate model of Cox, Ingersoll, and Ross (1985), a brute-force discretization leads to exploding binomial trees that do not recombine. The focus of this thesis, the huge influential generalized autoregressive conditional heteroskedastic (GARCH) model, is also bivariate.

Bollerslev (1986) and Taylor (1986) independently proposed the GARCH process for modeling the stochastic volatility of asset return.Since then, the model has been generalized and used extensively in the finance literature on the modeling of financial time series; see Bollerslev et al. (1992) and Engle and Patton (2001). As the model has received strong experimental support, its application to option pricing draws a lot of attention. Duan (1995) was the first to propose a GARCH option pricing model. The massive path dependency of the pricing model initially favors Monte Carlo simulation over trees. But the Monte Carlo estimate is probabilistic; furthermore, options that can be exercised early, the so-called American options, can be accurately priced only with simulation schemes that employ advanced techniques. The appearance of the trinomial tree of Ritchken and Trevor (1999) addresses these problems and makes a strong case for trees. Their algorithm is simple and claims to be accurate and efficient. It is also general enough to work beyond GARCH models. GARCH option pricing techniques that are not based on trees include the Markov chain approximation of Duan and Simonato (2001), the Edgeworth tree approximation of Duan et al. (2002), and analytical approximations as in Heston and Nandi (2000). Among them, only the Markov chain approximation approach is capable of handling American options.

This thesis will build on the methodology of the Ritchken-Trevor algorithm and its modified version by Cakici and Topyan (2000). From Lyuu and Wu (2003), we know that the Ritchken-Trevor-Cakici-Topyan (RTCT) option pricing algorithm is efficient only if it is restricted to a small n. Raising n to improve accuracy will quickly incur exponential slowdown. Worse, RTCT cannot grow beyond a certain maturity when explosion happens, which makes it useless for pricing derivatives with a longer maturity. Lyuu and Wu (2003) propose a new trinomial-tree GARCH option pricing algorithm that solves the above-mentioned weakness of RTCT. In the trinomial tree, the three successor nodes of each node must be such that their stock prices match the mean and variance of the model's stock price asymptotically. The three branches must carry valid branching probability. It is easy to know that the higher the volatility the wider the nodes are spaced to meet the requirements. The RTCT tree, like typical trinomial trees, takes a flat middle branch from

each node as in Fig. 1.1(a). The new tree of Lyuu and Wu (2003) departs from that by making the middle branch track the expected stock price as in Fig. 1.1(b). They call it the mean-tracking (MT) algorithm. By tracking the mean, the other two flanking branches are fanning out less in their attempt to match the mean and variance of the next stock price. This in turn yields more compact trees.

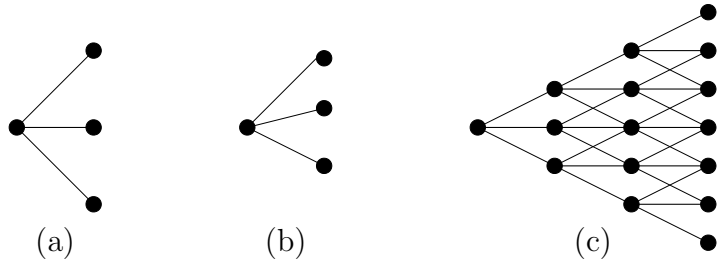

Figure 1.1: **Trinomial tree.**
(a) Three successor nodes follow each node. How widely the two flanking branches fan out around the middle branch depends on the volatility. (b) The middle branch may maintain a drift to minimize that width. (c) When the vertical node spacing is a constant, the number of nodes at any time t is $2t+1$. The total number of nodes of an N-period trinomial tree is $\sum_{t=0}^{N}(2t+1) = (N+1)^2 = O(N^2)$, a quadratic growth in maturity N.

Although the MT tree addresses the weaknesses of RTCT, it is still no problem free. When n increases, the computed option tend to drift from the true option value of Monte Carlo simulation. In order to mitigate the speed of this trend, we propose to use higher-order interpolation. Specifically, we will adopt cubic interpolation in the logarithmic domain.

The thesis is organized as follow. The GARCH model is presented in Section 2. Section 3 reviews the RTCT tree from which the MT tree derives. Several differences between them will be pointed out in this section. Section 4 covers backward induction with interpolated volatilities, which is needed to reduce the state space. In Section 5, we will present the MT option pricing algorithm and our improved version. Section 6 compares the performance of RTCT, the original MT tree and our improved MT tree. Section 7 concludes.

Chapter 2

The GARCH Model

Let S_t denote the asset price at date t and h_t the conditional volatility of the return over the period $[\,t, t+1\,]$ given the information at date t. Here, "one day" is just a convenient term for any elapsed time Δt. The following risk-neutral process for the logarithmic price $y_t \equiv \ln S_t$ is due to Duan (1995):

$$y_{t+1} = y_t + r - \frac{h_t^2}{2} + h_t \epsilon_{t+1} \tag{2.1}$$

where

$$\begin{aligned} h_{t+1}^2 &= \beta_0 + \beta_1 h_t^2 + \beta_2 h_t^2 (\epsilon_{t+1} - c)^2, \qquad (2.2) \\ \epsilon_{t+1} &\sim N(0,1) \text{ given information at date } t, \\ r &= \text{daily riskless return}, \\ c &\geq 0. \end{aligned}$$

The model is bivariate as its state is described by (y_t, h_t^2). Updating rule (2.2) for the squared volatility, due to Engle and Ng (1993), is also called the nonlinear asymmetric GARCH or NGARCH for short. Other GARCH models are surveyed in Duan (1997).

A positive c represents a negative correlation between the asset return and the volatility. We assume $\beta_0, \beta_1, \beta_2 \geq 0$ to make the squared volatility h_t^2 positive. We further impose $\beta_1 + \beta_2 < 1$ to make the model stationary. To price option based on the continuous-state GARCH model, we turn to trees with discrete states. But path dependence in the model will make the tree exponentially. The trinomial-tree approximation by Ritchken and Trevor relieves the explosion. Throughout the thesis, N will denote the maturity of the tree in days, which is also the maturity of the option to be priced by the tree.

Chapter 3

The RTCT Tree

The RTCT trinomial tree approximates the continuous-state GARCH process with discrete states as follow. Each state is represented as a node. Partition a day into n periods. Three states follow each state (y_t, h_t^2) after a period. As the trinomial tree recombines, $2n+1$ states at date $t+1$ follow each state at date t. Let $\gamma = h_0$ and $\gamma_n = \gamma/\sqrt{n}$. (The MT algorithm of Lyuu and Wu (2003) will choose a different γ.) The tree is laid over nodes that are spaced by γ_n in their logarithmic prices as depicted in Fig. 3.1. Consequently, the logarithmic price y_t on each node equals $y_0 + k\gamma_n$ for some integer k.

We next pick the jump size for state (y_t, h_t^2). The jump size determines how much the state's three successor states are spaced. As emphasized earlier, the magnitude of the jump size depends on the volatility h_t. By the geometry of the tree, the jump size must be some integer multiple η of γ_n. We call η the jump parameter. The jump parameter measures how much the two flanking branches fan out around the middle branch. It must be large enough for the three branches to match the mean and variance of y_{t+1}. The three nodes one period hence extend over $2\eta + 1$ nodes (inclusively), and inductively the $2n+1$ nodes from (y_t, h_t^2) at date $t+1$ extend over $2n\eta + 1$ nodes (inclusively). The larger the jump parameter η, the larger the tree because it extends over more nodes. The middle branch of the RTCT tree leaves the underlying asset's price unchanged. In contrast, the MT tree of Lyuu and Wu (2003) lets the middle branch track the mean of y_{t+1}, i.e., $y_t + r - (h_t^2/2)$. This idea will result in a smaller jump parameter η, thus yielding a more compact tree. Figure 3.2 illustrates a 1-day trinomial tree with each day partitioned into $n = 3$ periods.

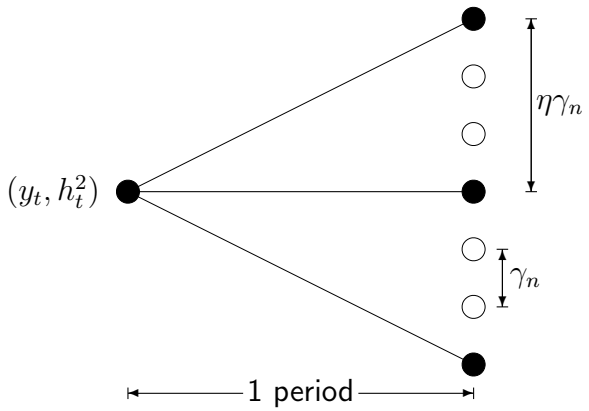

Figure 3.1: **Jump parameter η and jump size $\eta\gamma_n$.**
The tree is laid over a lattice whose vertically adjacent nodes are spaced by γ_n. The two flanking branches fan out around the middle branch to reach the two nodes that are η nodes away from the center. Although $2(\eta-1)$ hollow nodes are not reached from the node on the left, they may be reached from other nodes. Here $\eta = 3$.

The probabilities for the up, middle, and down branches are

$$p_u = \frac{h_t^2}{2\eta^2\gamma^2} + \frac{r - (h_t^2/2)}{2\eta\gamma\sqrt{n}}, \tag{3.1}$$

$$p_m = 1 - \frac{h_t^2}{\eta^2\gamma^2}, \tag{3.2}$$

$$p_d = \frac{h_t^2}{2\eta^2\gamma^2} - \frac{r - (h_t^2/2)}{2\eta\gamma\sqrt{n}}. \tag{3.3}$$

As the branching probabilities are picked to match the mean and variance of y_{t+1} given (y_t, h_t^2) asymptotically, the tree converges to the continuous-state model (2.1). From Eqs. (3.1)–(3.3), it is not hard to verify that valid branching probabilities exist (i.e., $0 \le p_u, p_m, p_d \le 1$) if and only if

$$\frac{|\, r - (h_t^2/2)\,|}{2\eta\gamma\sqrt{n}} \le \frac{h_t^2}{2\eta^2\gamma^2} \le \min\left(1 - \frac{|\, r - (h_t^2/2)\,|}{2\eta\gamma\sqrt{n}}, \frac{1}{2}\right). \tag{3.4}$$

We can dispense with the intermediate nodes between to create a $(2n+1)$-nomial tree (as shown in Fig. 3.3). The resulting model is multinomial with $2n+1$ branches from any state (y_t, h_t^2). These $2n+1$ successor nodes extend over $2n\eta+1$ nodes (inclusively). This multinomial tree is the building block of the RTCT tree.

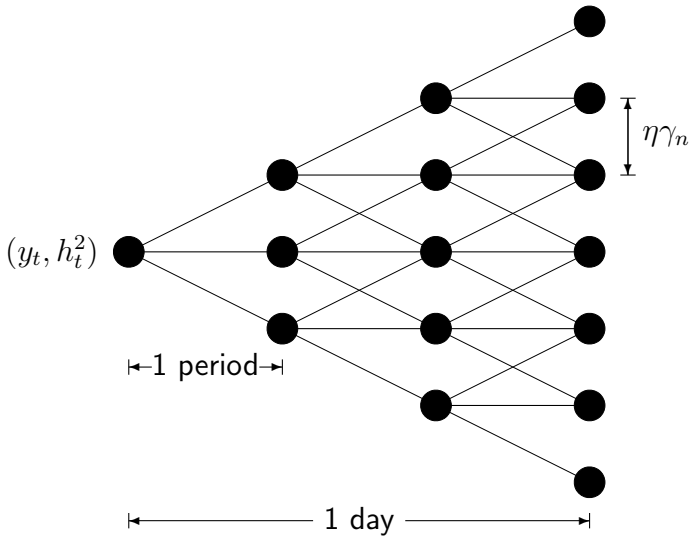

Figure 3.2: **RTCT trinomial tree for daily logarithmic price y_t for the duration of one day.**
A day is partitioned into $n = 3$ periods, and the jump size is $\eta\gamma_n$. The 7 states at date $t + 1$ approximate the probability distribution of y_{t+1} given (y_t, h_t^2).

Updating rule (2.2) must be modified to reflect the adoption of the discrete-states tree model. State (y_t, h_t^2) at date t is now followed by state $(y_t + \ell\eta\gamma_n, h_{t+1}^2)$ at date $t + 1$, where

$$\begin{aligned} h_{t+1}^2 &= \beta_0 + \beta_1 h_t^2 + \beta_2 h_t^2 (\epsilon'_{t+1} - c)^2, \\ \epsilon'_{t+1} &= \frac{\ell\eta\gamma_n - (r - h_t^2/2)}{h_t}, \\ \ell &= 0, \pm 1, \pm 2, \ldots, \pm n. \end{aligned} \tag{3.5}$$

We will call the resulting tree the *full* RTCT tree. For example,node A in Fig. 3.3 contains state (y_{t+1}, h_{t+1}^2), where

$$\begin{aligned} y_{t+1} &= y_t + (-2)\,\eta\gamma_n, \\ h_{t+1}^2 &= \beta_0 + \beta_1 h_t^2 + \beta_2 h_t^2 \left\{ \frac{(-2)\,\eta\gamma_n - [\,r - (h_t^2/2)\,]}{h_t^2} - c \right\}^2. \end{aligned}$$

As part of a larger RTCT tree, node A may contain other states which reach it from states other than (y_t, h_t^2).

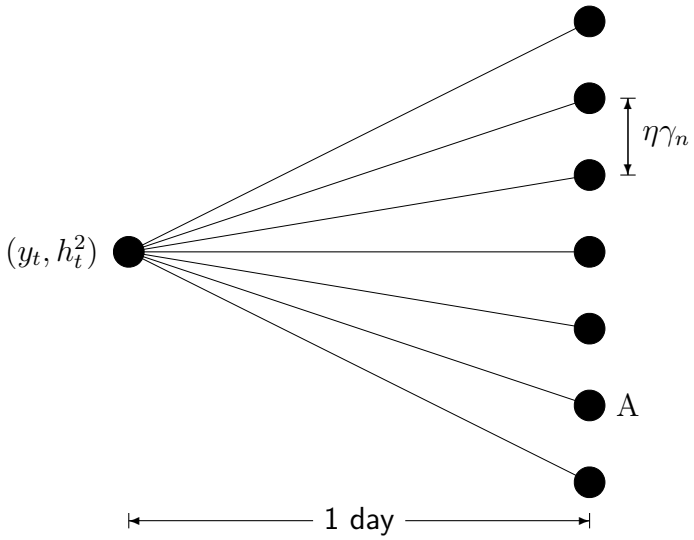

Figure 3.3: **RTCT multinomial tree for daily logarithmic price y_t for the duration of one day.**
This heptanomial tree is the outcome of the trinomial tree in Fig. 3.2 after removing its intraday nodes. Recall that $n = 3$. In general, there are $2n\eta+1$ nodes at date $t + 1$ between the top and bottom nodes (inclusive) spaced γ_n apart, but only $2n + 1$ of which are reachable from (y_t, h_t^2) and drawn above. The overwhelming majority of those $2n(\eta - 1)$ nodes not drawn are reached from the root state.

From the underlying trinomial model, the transition from state (y_t, h_t^2) to state $(y_t + \ell\eta\gamma_n, h_{t+1}^2)$ happens with probability

$$P(\ell) \equiv \sum_{j_u,j_m,j_d} \frac{n!}{j_u!j_m!j_d!}\, p_u^{j_u} p_m^{j_m} p_d^{j_d}, \tag{3.6}$$

where $j_u, j_m, j_d \geq 0$, $n = j_u + j_m + j_d$, and $\ell = j_u - j_d$. This seemingly complicated formula for probability $P(\ell)$ can be calculated very efficiently with the simple generating-function technique in Lyuu (2002) as follows. Note that

$$(p_u x + p_m + p_d x^{-1})^n = \sum_{\ell=-n}^{n} P(\ell)\, x^\ell.$$

Therefore, we can multiply out $(p_u x + p_m + p_d x^{-1})^n$ and then retrieve the probability $P(\ell)$ by reading off the coefficient of the power of x. The computation takes $O(n^2)$ steps. Compared with the complex formula (5.2), the generating function approach is straightforward and stable.

As volatility h_t changes through time, we may have to pick different jump parameter η for different states so that all branching probability p_u, p_m, and p_d lie between 0 and 1. This entails varying jump sizes. As the necessary requirement $p_m \geq 0$ implies

$$\eta \geq h_t/\gamma, \tag{3.7}$$

RTCT goes through

$$\eta = \lceil h_t/\gamma \rceil, \lceil h_t/\gamma \rceil + 1, \lceil h_t/\gamma \rceil + 2, \ldots \tag{3.8}$$

until valid probabilities are obtained or until their nonexistence is confirmed by inequalities (3.4). The latter case means the tree cannot grow further. When h_t^2 grows exponentially, the resulting tree must do likewise.

Every node at date t on the tree holds a different logarithmic price y_t. However, more than one path from the root state (y_0, h_0^2) may lead to the same node, each yielding a different squared volatility h_t^2. The number of possible values of h_t^2 at a node thus equals the number of paths reaching the node. Each h_t^2 picks its own jump parameter η. Figure 3.4 uses $n = 1$ to illustrate a three-day tree model. Node A and B each have one h_t^2, whereas node C has two. The h_t^2 at nodes A and B picks the same jump parameter $\eta = 2$. Node C could pick multiple jump parameter $\eta = 1, 2$. The overall tree structure will be irregular because of the varying jump parameter. Hollow node in Fig. 3.4 are not occupied because they are unreachable from the root state (y_0, h_0^2). If these squared volatilities pick different jump parameters such as those at node C and D, more than three branches will emanate from the node. More branches makes more nodes reachable.

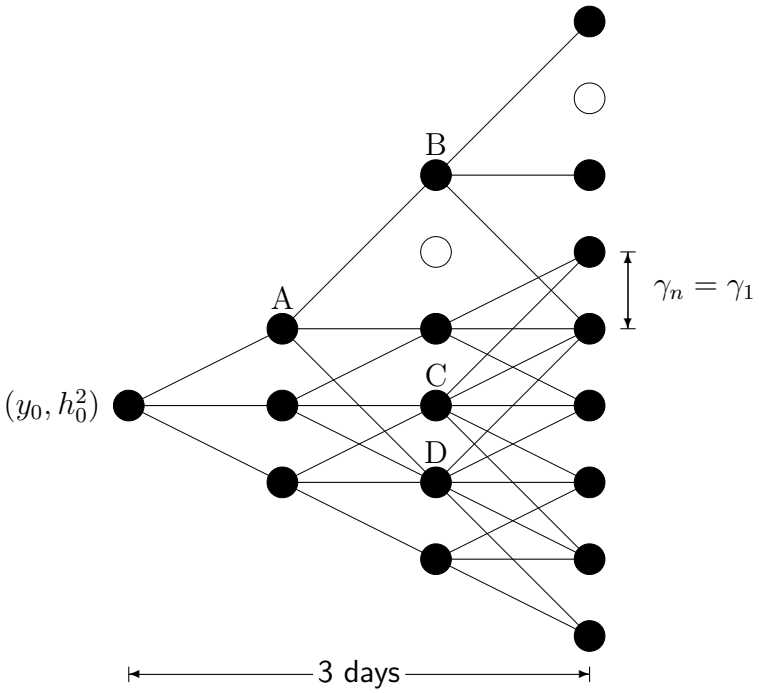

Figure 3.4: **Geometry of a 3-day RTCT tree.**
A day is partitioned into $n = 1$ period. Nodes A and B have a jump parameter of 2. Nodes C with two h_t^2 and D with three h_t^2 have two jump parameters: 1 and 2. All other nodes have a jump parameter of 1. Hollow nodes are not reachable from the root state (y_0, h_0^2).

Chapter 4

Interpolated Volatilities and Backward Induction

The number of possible volatilities h_t at a node equals the number of paths reaching it. Any algorithm that keeps all of these volatilities cannot be efficient as their count is exponential. Therefore, the full RTCT tree must be approximated. The standard approximation methodology by Hull and White (1993) and Ritchken, Sankarasubramanian, Vijh (1993) is adopted by both RTCT and MT. The RTCT tree keeps only the maximum volatility $h_{\max}$ and the minimum volatility $h_{\min}$ at each node. Then it creates $K-2$ volatilities between $h_{\min}$ and $h_{\max}$. We call these $K-2$ volatilities interpolated volatilities because they are not the results of actually applying updating rule (3.5). Instead, they are artificial volatilities generated via interpolation. Then there are K volatilities per node.

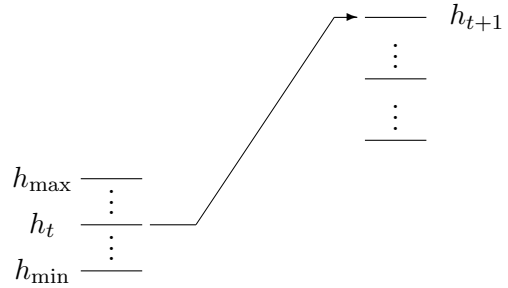

Figure 4.1: **Case where maximum volatility follows an interpolated volatility.**
The maximum volatility h_{t+1} at the node on the right follows interpolated volatility h_t.

At tree-building, RT calculates the $h_{\min}$ and $h_{\max}$ with these interpolated volatilities taken into consideration. Every node at date i generates $2n+1$ volatilities for each of its K volatilities via the updating rule. The number of volatilities generated per node is $K(2n+1)$. These volatilities at date i with their associated branches determine the $h_{\min}$ and $h_{\max}$ of every node at date $i+1$. For RT, $h_{\min}$ and $h_{\max}$ may be artificial if they might be the results of applying updating rule to interpolated volatilities of previous date (see Fig. 4.1 for illustration). On the other hand, CT takes a different route to calculate $h_{\min}$ and $h_{\max}$ at node. It calculates $h_{\min}$ and $h_{\max}$ in the tree-building process without these interpolated volatilities. Every node at date i generates $2n+1$ volatilities for each of its two volatilities $h_{\min}$ and $h_{\max}$ via the updating rule. The number of volatilities generated per node is hence only $2(2n+1)$. These volatilities at date i with their associated branches determine the $h_{\min}$ and $h_{\max}$ of every node at date $i+1$. But both CT and RT use interpolated volatilities in backward induction.

In RTCT, the K squared volatilities at a node are equally spaced between $h^2_{\min}$ and $h^2_{\max}$:

$$h^2_{\min} + j\,\frac{h^2_{\max} - h^2_{\min}}{K-1}, \quad j = 0, 1, 2, \ldots, K-1.$$

It is called the linear interpolation scheme. A different distribution will be used by MT algorithm in the tree-building process. To be specific, the K logarithms of squared volatilities are equally spaced between $\ln h^2_{\min}$ and $\ln h^2_{\max}$ in MT; they are

$$\exp\left[\ln h^2_{\min} + j\,\frac{\ln h^2_{\max} - \ln h^2_{\min}}{K-1}\right], \quad j = 0, 1, 2, \ldots, K-1.$$

Smaller volatilities are thus sampled more frequently than lager volatilities. We call it the log-linear interpolation scheme.

After the tree is built, backward induction commences. For the volatility h_{t+1} following state (y_t, h^2_t) via updating rule (3.5), RTCT will find the two volatilities that bracket h_{t+1}. The option price corresponding to h_{t+1} is then interpolated linearly from the option price corresponding to the bracketing volatilities. Figure 4.2 illustrates this procedure for one branch. After the option price from all $2n+1$ branches are available, the option price of state (y_t, h^2_t) is calculated as the average discounted value weighted by the branching probabilities. The above approximation is due to Hull and White (1993) and Ritchken, Sandkarasubramanian, and Vijh (1993).

CT has a serious problem that is not shared by RT. CT maintains only $h_{\min}$ and $h_{\max}$ at each node in growing the tree. The $K-2$ interpolated

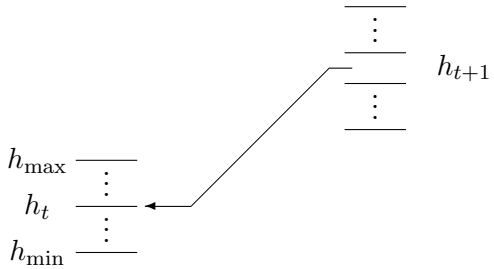

Figure 4.2: **Backward induction.**
Volatility h_{t+1} follows h_t by the updating rule. Because h_{t+1} does not match any interpolated volatility, its corresponding option value is found by interpolating from the two option values whose volatilities bracket it.

volatilities at each node are not involved in the tree-building process. Only in backward induction are these interpolated volatilities added to approximate full RTCT tree. It is therefore possible that an interpolated volatility's successor volatility reaches a node which is not reached during the tree-building process and thus has no option price at all. See Fig. 4.2 for illustration. If h_{t+1} lies in a node not reached at tree-building, the node will not have option values at all. Then backward induction cannot continue. Such rare situations do arise when n and N are both large. In contrast, RT takes the interpolated volatilities in building the tree and in backward induction. So RT does not have such a problem. MT follows RT in using all K volatilities (including $h_{\min}$, $h_{\max}$, and $K-2$ interpolated volatilities) of a node in growing the tree.

It is common practice to arise accuracy by increasing n. Unfortunately, the largest h_t grows exponentially in t when n exceeds the threshold $\beta_1+\beta_2 > 1$ as shown by Lyuu and Wu (2003). When this happens, the value of η will grows exponentially by relation (3.7). And when this happens, the RTCT tree will explode. And when the RTCT tree explodes, it cannot grow beyond a certain maturity.

Chapter 5

The Mean-Tracking (MT) Tree

The RTCT tree has at least four weaknesses. First, it explodes exponentially when n exceeds a threshold. Second, it is not known whether there is a simple formula for the threshold n^* so that the RTCT tree escapes explosion as long as $n \leq n^*$. Third, when explosion happens, the tree is cut short, making it unable to price derivatives with a longer maturity date. Fourth, option prices may fail to converge as n increases. The MT tree makes two changes to the RTCT tree. The first is to replace linear interpolation scheme with log-linear interpolation scheme. This addresses the convergence problem mentioned earlier. The second is to let the middle branch of multinomial tree track the mean of y_{t+1}. This addresses the explosion problem and its consequence of short maturity. This chapter will discuss variations of the MT tree to further enhance its performance.

5.1 Volatility Interpolation Schemes

The distribution of the volatilities plays an important role in pricing accuracy. RTCT assumes that the distribution is uniform: Interpolated volatilities are equally spaced between $h_{\min}$ and $h_{\max}$. It has been found that the actual distribution is closer to a lognormal distribution than a uniform distribution [25]. That means there are more interpolated volatilities located at lower values than at higher values. This is the reason for MT to adopt the log-linear interpolation scheme, in which the logarithmic volatilities are equally spaced. The log-linear scheme is also used by the Markov chain approximation of Duan and Simonato (2001) for the same numerical considerations. Similar findings in the case of the Asian option exist. For example, Dai (2004) proved that linear interpolation schemes result in overestimates, and Forsyth, et al. (2002) demonstrated that linear interpolation schemes may not converge to

the correct price.

5.2 Tree Building

At date t, let node A be the node closest to the mean of y_{t+1} given (y_t, h_t^2), which equals $y_t + r - (h_t^2/2)$. For convenience, we use μ to denote this conditional mean minus the current logarithmic price:

$$\mu \equiv r - \frac{h_t^2}{2}$$

(see Fig. 5.1). By the geometry of the tree, node A's logarithmic price equals $y_t + a\gamma_n$ for some integer a. The criterion by which node A is chosen ensures that

$$| a\gamma_n - \mu | \leq \frac{\gamma_n}{2}. \tag{5.1}$$

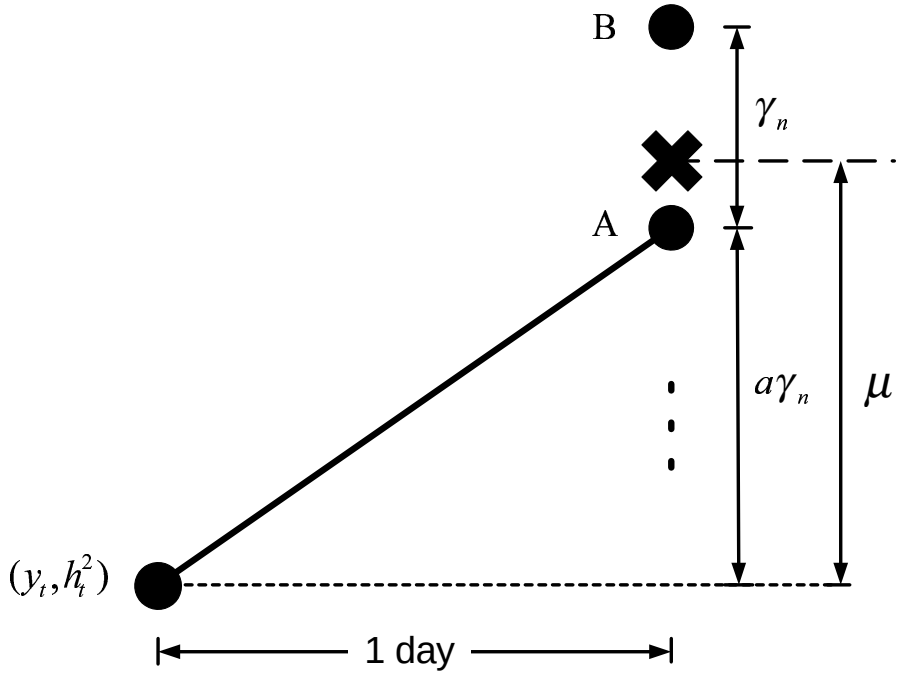

Figure 5.1: **The next middle node via mean tracking.**
The cross identifies the true mean of y_{t+1}. Two nodes, A and B, bracket it. Between them, node A has a logarithmic price closer to the mean. The number $a\gamma_n$ denotes the difference between y_t and node A's logarithmic price.

To create the desired multinomial tree, make the middle branch of the $(2n+1)$-nomial tree line up with node A as in Fig. 5.2. Although a node reaches only $2n+1$ nodes after day, the top and bottom node extend over

$$2n\eta + 1 \tag{5.2}$$

nodes as in RTCT. The probabilities for the upward, middle, and downward branches are

$$\begin{aligned} p_u &= \frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2} - \frac{a\gamma_n - \mu}{2n\eta\gamma_n}, \\ p_m &= 1 - \frac{nh_t^2 + (a\gamma_n - \mu)^2}{n^2\eta^2\gamma_n^2}, \\ p_d &= \frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2} + \frac{a\gamma_n - \mu}{2n\eta\gamma_n}. \end{aligned}$$

As they match the mean and variance of the GARCH process at date $t+1$ asymptotically, convergence in the limit is guaranteed. State (y_t, h_t^2) at date t is followed by state $(y_t + \ell\eta\gamma_n, h_{t+1}^2)$ at date $t+1$, where

$$\begin{aligned} h_{t+1}^2 &= \beta_0 + \beta_1 h_t^2 + \beta_2 h_t^2(\epsilon''_{t+1} - c)^2, \\ \epsilon''_{t+1} &= \frac{\ell\eta\gamma_n + a\gamma_n - (r - h_t^2/2)}{h_t}, \\ \ell &= 0, \pm 1, \pm 2, \ldots, \pm n. \end{aligned} \tag{5.3}$$

From the underlying trinomial model, this transition occurs with probability

$$\sum_{j_u, j_m, j_d} \frac{n!}{j_u! j_m! j_d!} p_u^{j_u} p_m^{j_m} p_d^{j_d},$$

where $j_u, j_m, j_d \geq 0$, $n = j_u + j_m + j_d$, and $\ell = j_u - j_d$.

The conditions for the probabilities to lie within 0 and 1, i.e., $0 \leq p_u, p_m, p_d \leq 1$, are

$$\frac{|\, a\gamma_n - \mu \,|}{2n\eta\gamma_n} \leq \frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2}, \tag{5.4}$$

$$\frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2} \leq \frac{1}{2}, \tag{5.5}$$

$$\frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2} \leq 1 - \frac{|\, a\gamma_n - \mu \,|}{2n\eta\gamma_n}. \tag{5.6}$$

Inequalities (5.4) -(5.5) are equivalent to

$$\frac{\sqrt{nh_t^2 + (a\gamma_n - \mu)^2}}{n\gamma_n} \leq \eta \leq \frac{nh_t^2 + (a\gamma_n - \mu)^2}{n\gamma_n |\, a\gamma_n - \mu \,|} \tag{5.7}$$

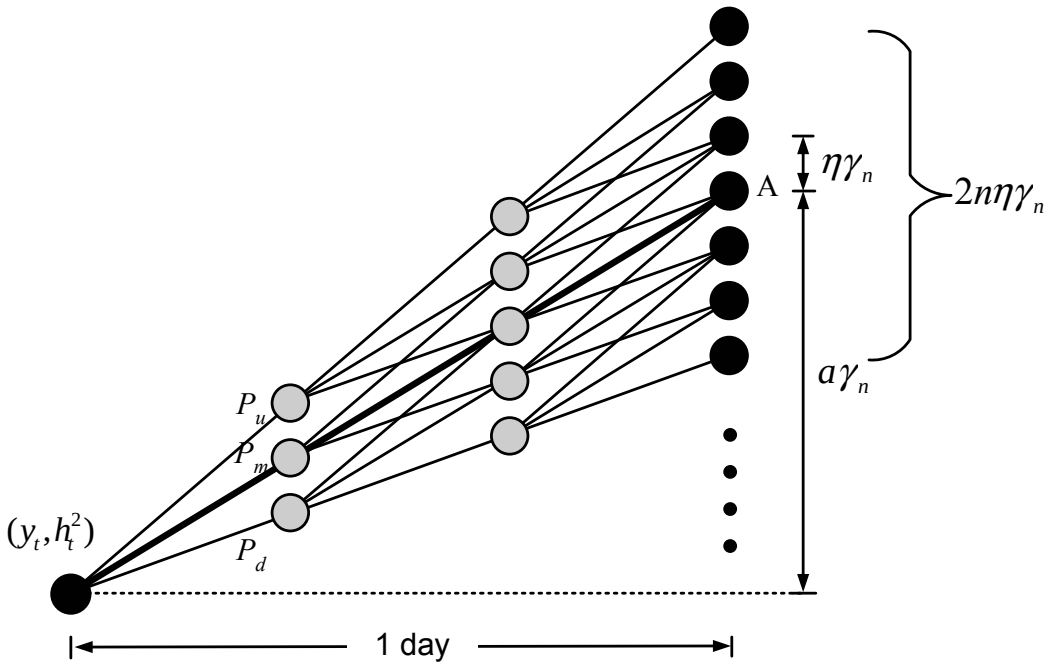

Figure 5.2: **The MT multinomial tree for daily logarithmic price y_t for a duration of one day.**
A day is partitioned into $n = 3$ periods, and the three jump sizes in each period are $(a\gamma_n/n) + \eta\gamma_n$ (upward), $a\gamma_n/n$ (middle), and $(a\gamma_n/n) - \eta\gamma_n$ (downward). The central branch of the tree lines up with node A, the node closest to the mean of y_{t+1} as stated in Fig. 5.1. The solid nodes are actually used in pricing, but the gray nodes are for illustration only. This heptanomial MT tree should be compared with the RTCT counterpart in Fig. 3.3.

Inequalities (5.1) and (5.5) together imply inequality (5.6) because

$$\frac{nh_t^2 + (a\gamma_n - \mu)^2}{2n^2\eta^2\gamma_n^2} + \frac{|a\gamma_n - \mu|}{2n\eta\gamma_n} \leq \frac{1}{2} + \frac{1}{4n\eta} \leq 1.$$

Hence the probabilities are valid if and only if the much simpler inequalities (5.7) hold.

MT can avoid the short-maturity problem of RTCT. Let $H_{\min}^2 \equiv \min(h_0^2, \beta_0/(1-\beta_1))$ to make $H_{\min}^2 \leq h_t^2$ for $t \geq 0$. Now, a valid integer η can always be found regardless of the value of n as long as γ_n is less than some constant. More precisely, interval (5.7) contains positive integers for the jump parameter η to take its value in when

$$\gamma_n^2 \leq H_{\min}^2. \tag{5.8}$$

With the existence of η guaranteed, MT will never be cut short. Rather than searching for an η to satisfy inequalities (5.7), MT simply sets

$$\eta = \left\lceil \frac{\sqrt{nh_t^2 + (a\gamma_n - \mu)^2}}{n\gamma_n} \right\rceil \tag{5.9}$$

although other choices are clearly possible, this particular choice is amenable to the analysis on the size of the MT tree.

We proceed to choose γ, hence γ_n as well because $\gamma_n = \gamma/\sqrt{n}$. If $\gamma \leq H_{\min}$, the γ_n satisfies inequality (5.8) for all n. A smaller γ generally leads to large trees, hence longer running times. On the other hand, a small γ is expected to result in better accuracy because of finer gain. To strike an overall balance between accuracy and convergence speed, we set $\gamma = H_{\min}/2$; thus

$$\gamma_n = \frac{H_{\min}}{2\sqrt{n}} \tag{5.10}$$

Now all the parameters of MT have been specified.

5.3 Backward Induction

Unlike RTCT, MT uses the log-linear interpolation scheme in backward induction, in which the K logarithms of squared volatilities are equally spaced between $\ln h_{\min}^2$ and $\ln h_{\max}^2$:

$$\exp\left[\ln h_{\min}^2 + j\,\frac{\ln h_{\max}^2 - \ln h_{\min}^2}{K-1} \right], \quad j = 0, 1, 2, \ldots, K-1. \tag{5.11}$$

Besides the simple log-linear scheme, MT can also adopt cubic interpolation over $\ln h_t^2$. Linear interpolation is easy and quick, but it may not be very precise. For linear interpolation, we just fit a straight line between two points, thus it will not catch any convexity/concavity of the underlying curve. We can usually reduce these errors by adding information from the points outside the two-point interval such as cubic interpolation. Moreover, we take the divided difference table method for cubic interpolation to maintain efficiency. It is faster than Lagrange interpolation because the computed node can be reused. The complexity of the divided difference table method is only $O(n^2)$, where n is the number of nodes which the interpolation uses. For convenience, we call MT using log-cubic interpolation MT-C and call MT using log-linear interpolation MT-LL. MT-C is similar to log-linear interpolation. The only difference is the way the volatilities are partitioned. In log-linear interpolation, the $K-2$ interpolated volatilities are linearly and

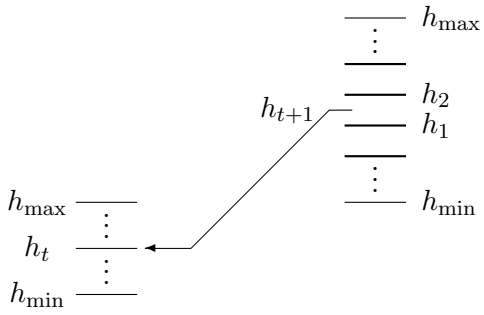

Figure 5.3: **Log-cubic interpolation.**
Volatility h_{t+1} follows h_t by the updating rule. Because h_{t+1} does not match any interpolated volatility, its corresponding option value is found by interpolating from the four option values whose volatilities bracket it.

equally spaced between $\ln h_{\min}^2$ and $\ln h_{\max}^2$ (see Eq. (5.11)). Then for the volatility h_{t+1} following state (y_t, h_t^2) via updating rule (5.3), we find two volatilities from these K logarithmic volatilities that bracket h_{t+1}. The option price corresponding to h_{t+1} is interpolated log-linearly from the option prices corresponding to the bracketing volatilities. In MT-C, we need four volatilities to get the estimated option price at h_{t+1}. Let the two volatilities bracketing h_{t+1} be h_1 and h_2 with $h_1 \leq h_2$. Besides h_1 and h_2, we need the volatility right below h_1 and the volatility right above h_2. The procedure is shown in Fig. 5.3. The option value corresponding to h_{t+1} is the result of log-cubic interpolation from these four interpolated volatilities. Under some circumstances, we cannot find four volatilities for use in cubic interpolation, such as when $h_1 = h_{\min}$ or $h_2 = h_{\max}$. When this happens, we replace log-cubic interpolation with log-linear interpolation.

Chapter 6

Comparison between MT-LL, MT-C and RTCT

In this section, we will investigate the performance of MT-C and compare it with other algorithms. For every n, we find that CT deviates from the results of Monte Carlo simulation (see Table 2). MT-C attains the same level of accuracy as MT-LL because the results produced by both are within the 95% confidence interval of Monte Carlo simulation of the full MT tree (5.3). Furthermore, MT-C provides option values close to the true option prices even with small n. Figure 6.1 is generated from Table 2. In Figure 6.1, CT deviates from the true option and the speed of downward trend accelerates when n increases. Compared with CT, the downward trend of MT-LL or MT-C is less pronounced. When the maturity is short, such as 2 or 5 days, the option prices of MT-C are the same as those produced by MT-LL. When n and/or N increase, the option values start to decrease. But the downward trend of MT-C is less than that of MT-LL as illustrated in Fig. 6.1. Hence MT-C attains better convergence than MT-LL.

All numerical data up to now assume $r = c = 0$. In Table 3, we review MT-C's accuracy under the GARCH option pricing model with nonzero r and c: $r = 5\%$ (annual) and $c = 0.5$ from Duan and Simonato (2001). Although a few of the computed option prices are outsides the 95% confidence interval, all option prices of MT-C are still close to the Monte Carlo estimates. Furthermore, they are as good as the the best computed option values in Table 3 of Duan and Simonato (2001) that are allowed the most computation times. So MT-C maintains the same accuracy as MT-LL in pricing options. From Tables 3 and 4, MT-C continues to enjoy better convergence than MT-LL when r and c are both nonzero.

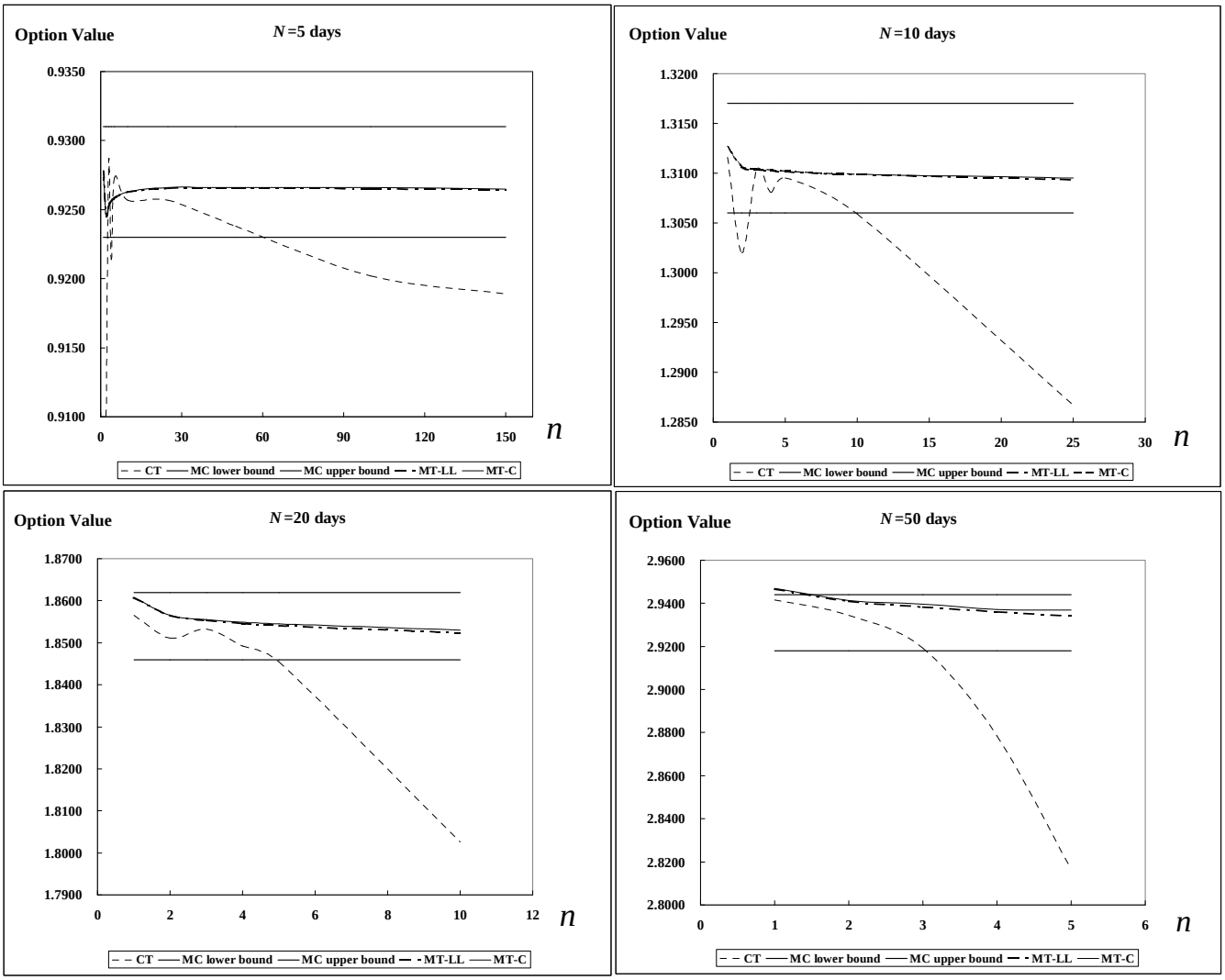

Figure 6.1: **Select option prices from Table 2**
. `MC lower bound` equals $^{\infty}L$; `MC upper bound` equals $^{\infty}U$.

Chapter 7

Conclusions

GARCH option pricing is difficult because of the GARCH model's bivariate and path-dependent character. The Ritchken-Trevor-Cakici-Topyan (RTCT) GARCH option pricing algorithm is inefficient as RTCT trees explode. Cakici and Topyan (2000) claimed that RTCT is accurate with $n = 1$ for vanilla options. Unfortunately, numerical data has demonstrated that both inaccuracy and explosion can result with such trees [25]. RTCT has been modified by Lyuu and Wu to obtain the mean-tracking (MT) tree. The MT tree is accurate and efficient when n does not exceed the simple threshold. This is the fist tree-based GARCH option pricing algorithm that will not explode if certain conditions are met. In the thesis, we modify MT to accelerate the convergence. We adopt log-cubic interpolation in backward induction. The methodology not only mitigates the trend to drift away from the true value but also does not lose accuracy. So MT-C, like MT proposed by Lyuu and Wu (2003), is an efficient algorithm for derivatives pricing under the GARCH option pricing model.

Bibliography

[1] Bollerslev, T. (1986) Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31, pp. 307-327.

[2] Bollerslev, T., Chou, R.Y., and Kroner, K. (1992) ARCH Modeling in Finance: a Review of the Theory and Empirical Evidence. *Journal of Econometricss*, 52, pp. 5-59.

[3] Cakici, N., and Topyan, K. (2000) The GARCH Option Pricing Model: a Lattice Approach. *Journal of Computational Finance*, 3(4), pp. 71-85.

[4] Cox, J.C., Ingersoll, J.E., and Ross, S.A. (1985) A Theory of Term Structure of Interst Rates. *Econometrica* 53(2), pp. 385-407.

[5] Cox, J.C., Ross, S.A., and Rubinstein, M. (1979) Option Pricing: a Simplified Approach. *Journal of Financial Economics*, 7(3), pp. 229-263

[6] Dai, T.-S. (2004) *Pricing Asian Options with Lattices.* Ph.D. Thesis. Department of Computer Science and Information Engineerinng, National Taiwan University, Taiwan.

[7] Dai, T.-S. (2004) An Exact Subexponential-Time Lattice Algorithm for Asian Options. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pp. 710-717, New Orleans, January 11-13, 2004. Philadelphia: Society for Industrial and Applied Mathematics, 2004.

[8] Dempster, M.A.H., and Richards, D.G. (2000) Pricing American Options Fitting the Smiles. *Mathematical Finance*, 10(4), pp. 157-177.

[9] Duan, J.-C. (1995) The GARCH Option Pricing Model. *Mathematical Finance*, 5(1), pp. 13-32.

[10] Duan, J.-C. (1997) Augmented GARCH(p, q)Process and Its Diffusion Limit. *Journal of Econometrics*, 79, pp. 97-127.

[11] Duan, J.-C., Gauthier, G., Sasseville, C., and Simonato, J.-G. (2003) Approximating American Option Prices in the GARCH Framework. *Journal of Futures Markets*, Vol. 23, pp. 915-929.

[12] Duan, J.-C., and Simonato, J.-G. (2001) American Option Pricing under GARCH by a Markov Chain Approximation. *Journal of Economic Dynamics & Control*, 22, pp. 1689-1718.

[13] Engle, R., and Ng, V. (1993) Measuring and Testing of the Impact of News on Volatility. *Journal of Finance*, 48, pp. 1749-1778.

[14] Heston, S.L., and Nandi, S. (2000) A Closed-Form GARCH Option Valuation Model. *Review of Financial Studies*, 13(3), pp. 585-625.

[15] Hull, J.C., and White, A. (1993a) Efficient Procedures for Valuing European and American Path-Dependent Options. *The Journal of Derivatives*, 1(1), pp. 21-31.

[16] Hull, J.C., and White, A. (1993b) One-Factor Intereat-Rate Models and the Valuation of Interet-Rate Derivative Securities. *Journal od Financial and Quantitive Analysis*, 28(2), pp. 235-254.

[17] Li, A., Ritchken, P., and Sankarasubramanian, L. (1995) Lattice Models for Pricing American Interest Rate Claims. *The Journal of Finance*, 50(2), pp. 719-739.

[18] Lyuu, Y.-D. (2002) *Financial Engineering & Computation: Principles, Mathematics, Algorithms. Cambridge*, U.K.: Cambridge University Press.

[19] Nelson, D.B., and Ramaswamy, K. (1990) Simple Binomial Processes as Diffusion Approximations in Financial Models. *Review of Financial Studies*, 3(3), pp. 393-430.

[20] Papadimitriou, C.H. (1995) *Computational Complexity*. Reading, MA: Addison-Wesley.

[21] Ritchken, P., and Sankarasubramanian, L. (1995) Volatility Structures of Forward Rates and the Dynamics of the Term Structure. *Mathmatical Finance*, 5(1), pp. 55-72.

[22] Ritchken, P., Sankarasubramanian, L., and Vijh, A.M. (1993) The Valuation of Path Dependent Contracts on the Average. *Managemet Science*, 39(10), pp. 1202-1213.

[23] Ritchen, P., and Trevor, R. (1999) Pricing Options under Generalized GARCH and Stochastic Volatility Processes. *Journal of Finance*, 54(1), pp. 377-402.

[24] Engle R. (2003) Risk and Volatility: Econometric Models and Finacial Practice. *Nobel Lecture*, December 8, 2003.

[25] Wu, C.-N. On Accurate and Provably Efficient GARCH Option Pricing Algorithms. MBA Thesis. Department of Finance, National Taiwan Unerversity, Taiwan, 2003.

K	2	10	20	50	100	200
Option price	4.2301	4.2365	4.2267	4.2274	4.2265	4.2268
${}^{\infty}L$	4.2714					
${}^{\infty}U$	4.3087					

Table 1: **Case where CT fails.**
K denotes the number of volatilities per node. The option is a European call with a strike price of 100 and a maturity of 100 days. ${}^{\infty}L$ and ${}^{\infty}U$ form the 95% confidence interval for the true option price based on Monte Carlo simulation of the continuous-state model (2.2) with 500,000 paths. The parameters are $S_0 = 100$, $r = 0$, $h_0^2 = 0.0001096$, $\gamma = h_0 = 0.010469$, $\beta_0 = 0.000007$, $\beta_1 = 0.9$, $\beta_2 = 0.04$, $n = 1$, and $c = 0$.

	Maturity of option (days)											
	2			5			10			20		
n	CT	MT-LL	MT-C	CT	MT-LL	MT-C	CT	MT-LL	MT-C	CT	MT-LL	MT-C
1	0.5888	0.5626	0.5626	0.9093*	0.9278	0.9278	1.3116	1.3126	1.3126	1.8565	1.8608	1.8608
2	0.5674	0.5799	0.5799	0.9091	0.9246	0.9246	1.3020	1.3107	1.3107	1.8511	1.8565	1.8565
3	0.5736	0.5833	0.5833	0.9284	0.9253	0.9253	1.3103	1.3104	1.3104	1.8532	1.8553	1.8555
4	0.5742	0.5845	0.5845	0.9214	0.9256	0.9256	1.3081	1.3103	1.3103	1.8492	1.8547	1.8549
5	0.5836	0.5851	0.5851	0.9273	0.9258	0.9258	1.3095	1.3102	1.3102	1.8454	1.8541	1.8545
10	0.5839	0.5864	0.5864	0.9257	0.9263	0.9263	1.3059	1.3099	1.3099			
25	0.5877	0.5872	0.5872	0.9257	0.9265	0.9266	1.2867	1.3093	1.3095			
50	0.5874	0.5874	0.5874	0.9238	0.9266	0.9266						
100	0.5876	0.5876	0.5876	0.9202	0.9265	0.9266						
150	0.5876	0.5876	0.5876	0.9189	0.9265	0.9265						
${}^{\infty}L$	0.5870			0.9230			1.3060			1.8460		
${}^{\infty}U$	0.5920			0.9310			1.3170			1.8620		

	50			75			100		
n	CT	MT-LL	MT-C	CT	MT-LL	MT-C	CT	MT-LL	MT-C
1	2.9415	2.9468	2.9469	3.6043	3.6105	3.6106	4.1647	4.1698	4.1715
2	2.9345	2.9410	2.9413	3.5976	3.6045	3.6052	4.1570	4.1640	4.1661
3	2.9193	2.9383	2.9397	3.5567	3.6009	3.6020			
4	2.8784	2.9362	2.9372						
${}^{\infty}L$	2.9180			3.5730			4.1420		
${}^{\infty}U$	2.9440			3.6050			4.1790		

Table 2: **Comparison between CT, MT-LL, and MT-C.**
${}^{\infty}DL$ and ${}^{\infty}DU$ form the 95% confidence interval for the true option price based on Monte Carlo simulation of the full MT tree (5.3) with 500,000 paths. None of the option prices of MT-LL and MT-C lie outside this confidence interval. ${}^{\infty}L$ and ${}^{\infty}U$ form the 95% confidence interval for the true option price based on Monte Carlo simulation of the continuous-state model (2.2) with 500,000 paths. The parameters are $S_0 = 100$, $r = 0$, $h_0^2 = 0.0001096$, $\gamma = h_0 = 0.010469$, $\beta_0 = 0.000006575$, $\beta_1 = 0.9$, $\beta_2 = 0.04$, $K = 20$, and $c = 0$.

	Maturity of option (days) = 30														
	X=55					X=50					X=45				
	K					*K*					*K*				
n	5	10	20	50	100	5	10	20	50	100	5	10	20	50	100
1	4.8283	4.8276	4.8275	4.8275	4.8275	1.1030	1.1042	1.1044	1.1045	1.1045	0.0748	0.0745	0.0745	0.0744	0.0744
2	4.8374	4.8343	4.8340	4.8340	4.8340	1.0878	1.0949	1.0956	1.0955	1.0955	0.0783	0.0763	0.0760	0.0760	0.0760
3	4.8433	4.8373	4.8360	4.8359	4.8359	1.0729	1.0895	1.0925	1.0927	1.0927	0.0816	0.0775	0.0765	0.0765	0.0765
∞L	4.8364					1.0862					0.0764				
∞U	4.8412					1.0898					0.0792				
DS	4.8377					1.0884					0.0715				

	Maturity of option (days) = 90														
	X=55					X=50					X=45				
	K					*K*					*K*				
n	5	10	20	50	100	5	10	20	50	100	5	10	20	50	100
1	4.9499	4.9517	4.9518	4.9520	4.9520	1.8306	1.8395	1.8397	1.8400	1.8400	0.4203	0.4217	0.4216	0.4217	0.4217
∞L	4.9505					1.8162					0.4131				
∞U	4.9587					1.8232					0.4185				
DS	4.955					1.8197					0.4036				

Table 3: **Accuracy of MT-C with nonzero** r **and** c**.**
The option is a European put with a strike price of X. ${}^{\infty}L$ and ${}^{\infty}U$ form the 95% confidence interval for the true option price based on Monte Carlo data from Duan and Simonato (2001). DS lists the option prices from Duan and Simonato (2001) given the most computational efforts. The table does not compute prices for $n > 1$ when the maturity exceeds 30 days because the tree explodes. All parameters are from Duan and Simonato (2001): $S_0 = 50$, $r = 5\%$ (annual), $h_0^2 = 0.0001096$, $\beta_0 = 0.00001$, $\beta_1 = 0.8$, $\beta_2 = 0.1$, and $c = 0.5$.

	Maturity of option (days) = 30											
	X=55				X=50				X=45			
	K				*K*				*K*			
n	10	20	50	100	10	20	50	100	10	20	50	100
1	4.8278	4.8576	4.8275	4.8275	1.1038	1.1043	1.1045	1.1045	0.0746	0.0745	0.0744	0.0744
2	4.8350	4.8342	4.8340	4.8340	1.0931	1.0950	1.0955	1.0955	0.0767	0.0762	0.0761	0.0760
3	4.8386	4.8366	4.8360	4.8359	1.0864	1.0912	1.0926	1.0927	0.0780	0.0769	0.0765	0.0765
${}^{\infty}L$	4.8364				1.0862				0.0764			
${}^{\infty}U$	4.8412				1.0898				0.0792			
DS	4.8377				1.0884				0.0715			

	Maturity of option (days) = 90											
	X=55				X=50				X=45			
	K				*K*				*K*			
n	10	20	50	100	10	20	50	100	10	20	50	100
1	4.9513	4.9519	4.9520	4.9520	1.8361	1.8393	1.8398	1.8399	0.4210	0.4215	0.4216	0.4216
${}^{\infty}L$	4.9505				1.8162				0.4131			
${}^{\infty}U$	4.9587				1.8232				0.4185			
DS	4.9550				1.8197				0.4036			

Table 4: **Accuracy of MT-LL with nonzero r and c.**
All parameter are the same as Table 3.