A General Computational Method for Calibration Based on Differential Trees

Yuh-Dauh Lyuu
Associate Professor
Dept. Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
Tel: 886-2-2362-5336 ext. 429
Fax: 886-2-2362-8167
lyuu@csie.ntu.edu.tw
http://www.csie.ntu.edu.tw/~lyuu

July 23, 1999

Abstract

This paper presents a general computational paradigm for model calibration called the differential tree method. The idea is very simple, and the method is particularly applicable when the model under consideration has a tree (lattice) structure. We illustrate its wide applicability with three canonical problems: no-arbitrage interest rate model calibration, spread and option-adjusted spread of non-benchmark bonds, and implied volatility of American-style options. Comprehensive computer experiments show the differential tree method to be highly efficient in all three case studies.

1 Introduction

Many computational problems in derivative pricing involve trees. The binomial option pricing model and various discrete-time interest rate models are familiar examples.

Computation on such structures conceptually proceeds via backward induction on the tree with local calculation at each node taking inputs from its successor nodes. Efficient algorithms are generally obtainable for pricing purposes if the tree recombines.

Our main concern, *model calibration*, is the inverse of the pricing problem: Given price information, it finds parameter values on the tree that generate *model prices* consistent with the data. Calibration, therefore, can be perceived as a root-finding problem. In this paper, we propose a general iterative procedure, the *differential tree method* (Lyuu [1995]), for the calibration of models based on trees.

The basic idea behind the differential tree method is simple. It takes advantage of the inductive structure of the tree to calculate not only the model price but also its derivatives with respect to the parameters whose values we desire. This extra effort adds only a small amount of overhead to the overall computation, yet it pays off immensely in terms of faster convergence and clean software design.

We illustrate the wide-ranging applications of the method with three canonical problems: calibration of no-arbitrage interest rate models (in this paper, the Black-Derman-Toy model [1990]), the calculation of spread and option-adjusted spread (OAS) of non-benchmark bonds, and the computation of the implied volatility of American-style options (all listed stock options in the U.S. are American, for example). In each case, we obtain efficient algorithms in terms of speed, computer memory, and convergence rates. This claim will be substantiated with extensive computer experiments.

This paper is organized as follows. In Section 2, the differential tree method is presented in its full generality. In Section 3, the method is applied to the calibration of the Black-Derman-Toy model. Treatments of spread and option-adjusted spread then follow suit in Section 4. Section 5 applies the method to calibrating the Cox-Ross-Rubinstein binomial option pricing model with American option prices. In each of the three cases, extensive computer experiments are conducted to investigate the method's performance and range of applicability. Section 6 ends the paper with conclusions.

2 The Differential Tree Method

Let $P(x)$ be a function defined by evaluation via backward induction on a tree. In our context, $P(x)$ denotes price. Such a framework captures derivative pricing un-

der discrete-time models. Its inverse function, the solution to $P(x) = p$ given p, defines the calibration problem. Pricing and calibration are therefore dual problems. In general, P and x can be vectored; for instance, $P(x)$ can be the discount function. To make matters simple, this paper focuses on binomial trees; generalization to multinomial trees is straightforward.

Backward induction on a tree can be seen as a cascading of function evaluations. Consider any node A with two successor nodes named B and C. In the chain of computations, node A computes a certain function $f_A(a, b, c)$ in which a is locally computable, whereas b and c are supplied by similar computation at nodes B and C, respectively. Node A in turn will feed its two predecessor nodes with f_A, and so on down the tree. Function P is simply the final result of these function evaluations at the root of the tree. If a, b, and c are functions of some parameter x, then f_A computed at A is also a function of x, and this holds inductively for P as well. See Figure 1's top row for illustration.

To solve for x, one needs a fast root-finding scheme. The Newton-Raphson method is one such scheme. This popular iterative method uses a function and its derivative to improve upon the current approximation to the root until the error lies within some specified bounds. Being quadratically convergent, the Newton-Raphson method doubles the precision in each iteration.

To employ the Newton-Raphson method to solve $P(x) = p$, one needs $P'(x)$. In principle, we can write down $P(x)$ explicitly and then differentiate it term by term. This idea becomes infeasible, however, when the tree is large. Fortunately, the differential tree method takes advantage of the tree by computing $P'(x)$ along with $P(x)$ inductively, as described below.

The node A mentioned before computes df_A/dx in addition to f_A and then forwards both f_A and df_A/dx to its two predecessor nodes. This extra calculation is straightforward: With db/dx and dc/dx from nodes B and C, respectively (besides $b(x)$ and $c(x)$ generated for the computation of $P(x)$, of course), and with da/dx available locally, we can calculate df_A/dx thus,

$$\frac{df_A(a(x), b(x), c(x))}{dx} = \frac{\partial f_A}{\partial a}\frac{da(x)}{dx} + \frac{\partial f_A}{\partial b}\frac{db(x)}{dx} + \frac{\partial f_A}{\partial c}\frac{dc(x)}{dx}.$$

Backward induction now continues from node A to its two predecessor nodes and, inductively, toward the root, which produces $P(x)$ and $P'(x)$. The Newton-Raphson method then finds a new approximation to the root before the next iteration begins. This process repeats itself until the desired accuracy is achieved. The overall scheme is

simple and adds little overhead to the original backward induction algorithm. We will call a tree with these derivative values a *differential tree* (see Figure 1 for illustration).

In general, x can be a vector. Take the two-dimensional case as an example, $\boldsymbol{x} \equiv (s,t)$. Then we need to calculate (Hildebrand [1976])

$$\begin{aligned}
\frac{\partial f_A(a(\boldsymbol{x}),b(\boldsymbol{x}),c(\boldsymbol{x}))}{\partial s} &= \frac{\partial f_A}{\partial a}\frac{\partial a(\boldsymbol{x})}{\partial s} + \frac{\partial f_A}{\partial b}\frac{\partial b(\boldsymbol{x})}{\partial s} + \frac{\partial f_A}{\partial c}\frac{\partial c(\boldsymbol{x})}{\partial s} \\
\frac{\partial f_A(a(\boldsymbol{x}),b(\boldsymbol{x}),c(\boldsymbol{x}))}{\partial t} &= \frac{\partial f_A}{\partial a}\frac{\partial a(\boldsymbol{x})}{\partial t} + \frac{\partial f_A}{\partial b}\frac{\partial b(\boldsymbol{x})}{\partial t} + \frac{\partial f_A}{\partial c}\frac{\partial c(\boldsymbol{x})}{\partial t}
\end{aligned}$$

Again, inductively, $\partial b(\boldsymbol{x})/\partial s$, $\partial c(\boldsymbol{x})/\partial s$, $\partial b(\boldsymbol{x})/\partial t$, and $\partial c(\boldsymbol{x})/\partial t$ are available from nodes B and C, and the same computational principle applies. In the general case where $\boldsymbol{x}$ is n-dimensional, the differential tree has to calculate n partial derivatives.

Let us illustrate the differential tree idea with an example: implied volatility under the binomial option model. Consider European puts. The standard CRR binomial pricing model implies the following backward induction formulation (Hull [1997]),

$$f_A(\sigma,b,c) = \begin{cases} \frac{pb+(1-p)c}{e^{r\Delta t}} & \text{if A is an internal node} \\ \max(X - Su^j, 0) & \text{if A is a terminal node} \end{cases}$$

Here,

$$p \equiv \frac{e^{r\Delta t} - d}{u - d}$$

is the probability of an up move, r is the interest rate, S is the current stock price, $u \equiv e^{\sigma\sqrt{\Delta t}}$, $d = 1/u$, Su^j is the stock price at node A (so the number of up moves exceeds that of down moves by j in order to reach A), X is the strike price, and a time period has duration Δt. To apply the differential tree method, simply compute the following derivative as well,

$$\frac{df_A}{d\sigma} = \begin{cases} \frac{p'b+pb'-p'c+(1-p)c'}{e^{r\Delta t}} & \text{if A is an internal node} \\ 0 & \text{if A is a terminal node but not exercised} \\ -j\sqrt{\Delta t}\,Su^j & \text{if A is a terminal node but exercised} \end{cases}$$

The sections to follow deal with three canonical calibration problems: interest rate model calibration, spread and option-adjusted spread of non-benchmark bonds, and implied volatility of American options (described above). All our programs are written in C. Following Patterson and Hennessy [1994], we take the position that the running time is the only valid performance measure. Other measures such as the number of iterations and/or the number of time partitions needed for convergence

will be mentioned but not given dominating emphasis since they lead to misleading conclusions (Lyuu [1998]).

Before closing this section, a few words about the computer setup and performance reports. All timing information will be in seconds. A variety of computing platforms were employed, including the Intel Pentium MMX processor, a 75MHz Sun SPARCstation 20, and a 200MHz Sun UltraSPARC. (The 300MHz Sun UltraSPARC IIi with a good C compiler from Sun was capable of cutting the reported running times based on the Sun SPARCstation 20 by about 90%.) Throughout this paper, n denotes the depth of the tree, i.e., the total number of time periods.

3 Calibration of Interest Rate Models: the Black-Derman-Toy Model

Our first application is concerned with the calibration of interest rate models. Calibration aims to set up a short rate tree consistent with the observed term structures, specifically the yields and, in some models, yield volatilities of zero-coupon bonds of all maturities as well. Although the differential tree method clearly applies to other models, we pick the Black-Derman-Toy (BDT) model for two reasons: analytical intractability and popularity. Calibration of the BDT model needs to fit both the bond prices and their yield volatilities.

For the BDT binomial tree, the logarithm of the future short rate obeys the binomial distribution. The limiting distribution for the short rate is hence lognormal. The BDT binomial tree of possible short rates for each future period is constructed as follows. Each short rate is followed by two short rates for the following time period of length Δt. In Figure 2, node A coincides with the start of period j during which short rate r is in effect. At the conclusion of period j, a new short rate goes into effect for period $j+1$. This may take one of two possible values: r_l, the "low" short rate outcome (up move, for the bond price), shown at node B, or r_h, the "high" short rate outcome (down move, for the bond price), at node C. Each rate has a fifty percent chance of occurring in a risk-neutral economy. Percent volatility of the short rate, $\Delta r/r$, is

$$\sigma = \frac{1}{2\sqrt{\Delta t}} \times \ln(r_h/r_l)$$

when the short rate follows a lognormal process in the limit. Note that

$$\frac{r_h}{r_l} = e^{2\sigma\sqrt{\Delta t}}. \tag{1}$$

To nail down the values of r_h and r_l, we need information from the current term structures to establish the relation between r and its two successors.

As the binomial process unfolds, we make sure that the paths recombine. In general, there are j possible rates applicable for period j. They are

$$r_j,\ r_j v_j,\ r_j v_j^2,\ \ldots,\ r_j v_j^{j-1},$$

where

$$v_j = e^{2\sigma_j\sqrt{\Delta t}}$$

is the *multiplicative ratio* for the rates in period j. The volatilities σ_j above are indexed by j because volatility is a function of time. We shall call r_j the *baseline rate*. Figure 3 depicts the tree structure.

With the abstract process in place, concrete numbers are needed to set it in motion: They are the annualized rates of return associated with the various riskless bonds making up the benchmark yield curve and their volatilities. For example, the on-the-run yield curve may be used as the benchmark curve. The term structure of volatilities (or volatility structure) may be estimated either from historical data or from interest rate derivatives. The binomial tree should be consistent with both term structures. From now on, for economy of expression, all numbers are measured by the period ($\Delta t = 1$) unless otherwise stated.

To store the whole tree, the space requirement would be proportional to n^2, which is prohibitively expensive for a large tree. For instance, modeling daily interest rate movements for thirty years amounts to keeping an array of roughly $(30 \times 365)^2/2 \approx 6 \times 10^7$ double-precision floating-point numbers. If each number takes up eight bytes, storage alone would consume nearly half a gigabyte! Fortunately, memory requirement can be made minimal. This is because only the baseline rates r_i's and the multiplicative ratios v_i's need to be stored in computer memory since the rest of the tree are known functions of them. The storage requirement is hence down to $O(n)$.

See Sandmann and Sondermann [1994] for the pitfall in using continuously compounded rates. Canabarro [1995] and Backus, Foresi, and Zin [1996] assess the BDT model critically. See Ingersoll [1987] for a history of early models. An influential methodology pioneered by Ho and Lee [1986] takes the market yield curve as given.

Models based on such a paradigm are usually called *no-arbitrage models*, to which the BDT model belongs.

3.1 The differential tree method

To derive the actual values for r_h and r_l, a naive approach starts with the implied one-period forward rates and their volatilities. It then dictates that the expected short rate equal the implied forward rate for the same period; in other words, the *unbiased expectations theory* holds. This method corresponds to the scheme "with no iteration" in Bjerksund and Stensland [1996]. However, Lyuu [1995] has proved that this approach overestimates the prices of benchmark securities as long as the yield volatilities are positive, independent of whether the volatility structure is fit.

We now present a correct approach using the differential tree method. Recall that there are j possible short rates for period j. According to the binomial interest rate model, the j rates are $r_j, r_j v_j, r_j {v_j}^2, \ldots, r_j {v_j}^{j-1}$, where r_j is the baseline rate. Suppose the price of the j-period zero moves up to P_u and down to P_d one period from today. Obviously, P_u and P_d are functions of r_j and v_j. In a risk-neutral world, it must hold that

$$\frac{\frac{1}{2}P_u + \frac{1}{2}P_d}{1 + r_1} = \frac{1}{(1+y)^j}, \tag{2}$$

where y is today's yield of the j-period zero, which is known.

Viewed from now, the future $(j-1)$-period yield at the beginning of period two is uncertain. Let y_u represent the $(j-1)$-period yield to maturity at the "up" node, y_d the $(j-1)$-period yield to maturity at the "down" node, and κ^2 the variance viewed from now of the $(j-1)$-period yield to maturity. The variance of the $(j-1)$-period yield is

$$\kappa^2 = p\,(1-p)\ln^2(y_u/y_d).$$

Hence, for $p = 1/2$,

$$\kappa = (1/2)\ln(y_u/y_d).$$

As the bonds are zero-coupon bonds,

$$y_u = {P_u}^{-\frac{1}{j-1}} - 1 \quad \text{and} \quad y_d = {P_d}^{-\frac{1}{j-1}} - 1.$$

Substitute to get

$$\kappa = (1/2)\ln\left(\frac{{P_u}^{-\frac{1}{j-1}} - 1}{{P_d}^{-\frac{1}{j-1}} - 1}\right). \tag{3}$$

Finally, rearrange (2) and (3) as simultaneous equations,

$$\begin{aligned} f(P_u, P_d) &\equiv P_u + P_d - \frac{2\,(1+r_1)}{(1+y)^j} = 0 \\ g(P_u, P_d) &\equiv {P_u}^{-\frac{1}{j-1}} - 1 - e^{2\kappa}\left({P_d}^{-\frac{1}{j-1}} - 1\right) = 0 \end{aligned}$$

Since P_u and P_d are functions of r_j and v_j, $f(P_u, P_d)$ and $g(P_u, P_d)$ are functions of r_j and v_j as well, denoted $F(r_j, v_j)$ and $G(r_j, v_j)$, respectively. For economy of expression, we use $f(r, v)$ instead of $f(r_j, v_j)$, $g(r, v)$ instead of $g(r_j, v_j)$, and so on. Given the kth approximation $(r(k), v(k))$, the Newton-Raphson method says the $(k+1)$st approximation $(r(k+1), v(k+1))$ satisfies

$$\begin{bmatrix} \frac{\partial F(r(k),v(k))}{\partial r} & \frac{\partial F(r(k),v(k))}{\partial v} \\ \frac{\partial G(r(k),v(k))}{\partial r} & \frac{\partial G(r(k),v(k))}{\partial v} \end{bmatrix} \begin{bmatrix} \Delta r(k+1) \\ \Delta v(k+1) \end{bmatrix} = - \begin{bmatrix} F(r(k), v(k)) \\ G(r(k), v(k)) \end{bmatrix}$$

where $\Delta r(k+1) \equiv r(k+1) - r(k)$ and $\Delta v(k+1) \equiv v(k+1) - v(k)$.

We still need $\partial F/\partial r$, $\partial F/\partial v$, $\partial G/\partial r$, and $\partial G/\partial v$ in the matrix above to solve for $(r(k+1), v(k+1))$. Obviously,

$$\begin{aligned} \frac{\partial F}{\partial r} &= \frac{\partial P_u}{\partial r} + \frac{\partial P_d}{\partial r} \\ \frac{\partial F}{\partial v} &= \frac{\partial P_u}{\partial v} + \frac{\partial P_d}{\partial v} \end{aligned}$$

and, by the chain rule,

$$\begin{aligned} \frac{\partial G}{\partial r} &= \frac{\partial g}{\partial P_u}\frac{\partial P_u}{\partial r} + \frac{\partial g}{\partial P_d}\frac{\partial P_d}{\partial r} \\ \frac{\partial G}{\partial v} &= \frac{\partial g}{\partial P_u}\frac{\partial P_u}{\partial v} + \frac{\partial g}{\partial P_d}\frac{\partial P_d}{\partial v} \end{aligned}$$

In the above four equations, the common terms requiring evaluations are $\partial P_u/\partial r$, $\partial P_d/\partial r$, $\partial P_u/\partial v$, $\partial P_d/\partial v$, $\partial g/\partial P_u$, and $\partial g/\partial P_d$. The differential tree method can compute them as follows. Working backward, the method finds P_u, P_d, $\partial P_u/\partial r$, $\partial P_d/\partial r$, $\partial P_u/\partial v$, and $\partial P_d/\partial v$. As for the two remaining terms $\partial g/\partial P_u$ and $\partial g/\partial P_d$, they can be computed directly from the definition of g as

$$\begin{aligned} \frac{\partial g}{\partial P_u} &= -\frac{1}{j-1}{P_u}^{-\frac{j}{j-1}} \\ \frac{\partial g}{\partial P_d} &= e^{2\kappa}\frac{1}{j-1}{P_d}^{-\frac{j}{j-1}} \end{aligned}$$

Thus we conclude the matrix can be set up.

The standard differential tree method leads to a backward induction implementation. Since each pass across the tree takes $O(j^2)$ time to obtain the jth baseline rate and the jth multiplicative ratio, the total time is $O(n^3)$, assuming the Newton-Raphson method takes a constant number of iterations to get to the desired accuracy. This assumption turns out to be reasonable in practice, as we will see later.

3.2 Forward induction

To reduce the computation time to $O(n^2)$, forward induction can be employed (Jamshidian [1991]). This scheme inductively figures out, by moving forward in time, how much \$1 at a node contributes to the total price (see Figure 4). This number is called the *state price* since it is the price of a claim that pays \$1 at that particular state (node) and zero elsewhere. We call a tree with these state prices a *binomial state price tree*. As before, there is no need to explicitly store the whole tree.

Let us be more precise about the mechanism. Suppose we are at the end of period j. So there are $j+1$ nodes. Let the baseline rate for period j be $r_j = r$, the multiplicative ratio be $v_j = v$, and $P_1, \ldots, P_j$ be the state prices at the nodes of the beginning of period j. One dollar j periods from now has a known market value of $1/(1+y)^j$, where y denotes the j-period spot rate. Alternatively, one dollar at the end of period j has a present value of

$$f(r,v) \equiv \frac{P_1}{1+r} + \frac{P_2}{1+rv} + \frac{P_3}{1+rv^2} + \cdots + \frac{P_j}{1+rv^{j-1}}.$$

To match the yield volatilities, we use the same logic as leads to (3) to obtain

$$g(r,v) \equiv (1/2)\ln\left(\frac{\left(\frac{P_{u,1}}{1+rv} + \frac{P_{u,2}}{1+rv^2} + \cdots + \frac{P_{u,j}}{1+rv^{j-1}}\right)^{-\frac{1}{j-1}} - 1}{\left(\frac{P_{d,1}}{1+r} + \frac{P_{d,2}}{1+rv} + \cdots + \frac{P_{d,j-1}}{1+rv^{j-2}}\right)^{-\frac{1}{j-1}} - 1}\right).$$

In the above the $P_{u,i}$'s denote the state prices of the tree rooted at the up node, while the $P_{d,i}$'s denote the state prices of the tree rooted at the down node. Now, solve

$$\begin{aligned} f(r,v) &= \frac{1}{(1+y)^j} \\ g(r,v) &= \kappa \end{aligned}$$

Here, the Newton-Raphson method can be used to solve for r and v as the derivatives are easy to evaluate. The overall running time is now $O(n^2)$. We emphasize

that forward induction is an efficient implementation of backward induction with differential trees; it does not change the underlying computational logic a bit.

3.3 Experimental results

Consider the following term structures in Hull and White [1990],

$$\begin{array}{ll} r^* + 0.05 \times \ln t & \text{for } t\text{-year zero-coupon bond yield} \\ 1.4 \times \left(1 - e^{-0.1 \times t}\right) / t & \text{for } t\text{-year zero-coupon bond yield volatility} \end{array}$$

3.3.1 Convergence of the differential tree method

The differential tree method converges very fast. Figure 5 reveals that, with one partition per year, it takes an average of less than 3.5 iterations to achieve a relative error of 10^{-13} with $r^* = 0.06$. With the Newton-Raphson method, it is critical to get a suitable initial guess. Using the baseline rate and multiplicative ratio of the previous period as the initial guesses for the current period has turned out to work well.

3.3.2 Comparison between forward induction and backward induction

The running time of backward induction with the differential tree is $O\left(n^3\right)$ versus $O\left(n^2\right)$ for forward induction. We benchmark backward induction with differential trees and forward induction in Figure 6. The result is as expected: The forward induction implementation is far more efficient. The running time is quadratic at $0.000020 \times n^2$ seconds. Furthermore, we can calibrate the tree up to 270,000 years (see Figure 7). From here on, the paper adopts a relative error of 10^{-6} since our emphasis now shifts toward the growth rate of running time.

3.3.3 Aspects of forward induction

Figure 8 looks at the problem from a different perspective. It fixes the time span at 30 years and looks at how forward induction performs by increasing the number of partitions. The conclusion is it can go as fine as 70 periods per year. The running time is quadratic, at about $0.000098 \times n^2$ seconds. The experiment is then repeated for 10-year trees with the results shown in Figure 9. The running time there grows roughly as $0.000051 \times n^2$ seconds, and we can go as fine as 1,850 periods per year.

4 Spread Calculation

4.1 Spread of non-benchmark option-free bonds

Model prices calculated from trees calibrated off the benchmark bonds as a rule do not match market prices of non-benchmark bonds. To gauge the incremental return, or *yield spread*, over the benchmark bonds, one looks for the spread over the short rates in the tree that equates the model price and the market price (Fabozzi [1991]). When the underlying bond contains embedded options, the spread is called *option-adjusted spread*. We treat option-free bonds first and bonds with embedded options later. The techniques are identical save for the possibility of early exercise.

When a constant amount s is added to every rate in the binomial interest rate tree, the model price will be a monotonically decreasing, convex function of s. Call this function $P(s)$. For a given market price p, we can employ a root-finding algorithm to solve $P(s) = p$.

Evaluating $P'(s)$ directly by expansion looks intimidating. Fortunately, the differential tree method can be used to evaluate both $P(s)$ and $P'(s)$ in one pass. Here is the idea. Every node of the tree, call it node A, is the root of a subtree of the original tree and computes a price $p_A(s)$ during the process of computing the model price $P(s)$ via backward induction. Node A is also associated with a short rate r. Hence, prices computed at A's two successor nodes, B and C, will be discounted by the factor $1/(1+r+s)$ to obtain $p_A(s)$,

$$p_A(s) = c + \frac{p_B(s) + p_C(s)}{2\,(1+r+s)}.$$

All this is standard. To compute $p'_A(s)$ as well, node A simply calculates

$$p'_A(s) = \frac{p'_B(s) + p'_C(s)}{2\,(1+r+s)} - \frac{p_B(s) + p_C(s)}{2\,(1+r+s)^2},$$

where c denotes the cash flow at A. Now, computing $p'_A(s)$ is easy since $p'_B(s)$ and $p'_C(s)$ have been supplied by nodes B and C. Applying the above argument inductively will eventually lead to $P(s)$ and $P'(s)$. See Figure 10 for illustration.

The running time depends on the convergence rate of the Newton-Raphson method. In practice, only a small number of iterations is needed to bring the algorithm to convergence. Hence the empirical running time is $O(n^2)$. The memory requirement is again linear in n.

The experiments apply the differential tree method to zero-coupon bonds under the BDT model and flat term structures,

$$\begin{array}{ll} 8\% & \text{for } t\text{-year zero-coupon bond yield} \\ 10\% & \text{for } t\text{-year zero-coupon bond yield volatility} \end{array}$$

The differential tree method converges very fast. For example, it takes about 7.85 seconds for a tree with 500 periods on a 75MHz Sun SPARCstation 20 with 64MB of DRAM. In general, the running time is about $0.000031 \times n^2$ seconds.

4.2 Option-adjusted spread of non-benchmark bonds with embedded options

The calculation of OAS is the same as before except for the possibility of early exercise. The function $p_A(s)$ is slightly modified,

$$p_A(s) = c + \min\left(C, \frac{p_B(s) + p_C(s)}{2\,(1 + r + s)}\right), \tag{4}$$

where C is the call price. The derivative $p'_A(s)$ is then

$$p'_A(s) = \begin{cases} 0 & \text{if exercised} \\ \frac{p'_B(s)+p'_C(s)}{2\,(1+r+s)} - \frac{p_B(s)+p_C(s)}{2\,(1+r+s)^2} & \text{if not exercised} \end{cases}$$

The experiments apply the differential tree method to 8% callable coupon bonds under the BDT model and flat term structures. The results show that the differential tree method converges very fast. For example, it takes about 8.11 seconds for a tree with 500 periods on a 75MHz Sun SPARCstation 20 with 64MB of DRAM. In general, the running time is quadratic, at about $0.0000065 \times n^2$ seconds per iteration.

5 Volatility Implied by American Options

The last case study is concerned with the implied volatility of American options. With early exercise, the option price as a function of volatility σ is no longer differentiable everywhere. The backward induction formulation for American puts is

$$f_A(\sigma, b, c) = \begin{cases} \max\left(\frac{pb+(1-p)\,c}{e^{r\Delta t}}, X - Su^j\right) & \text{if A is an internal node} \\ \max\left(X - Su^j, 0\right) & \text{if A is a terminal node} \end{cases}$$

(Compare it to European options in Section 2.) To apply the differential tree method, compute

$$\frac{df_A}{d\sigma} = \begin{cases} \frac{p'b+pb'-p'c+(1-p)\,c'}{e^{r\Delta t}} & \text{if A is an internal node but not exercised} \\ 0 & \text{if A is a terminal node but not exercised} \\ -j\sqrt{\Delta t}\,Su^j & \text{if A is exercised} \end{cases} \tag{5}$$

Note that the computational problem here is structurally similar to the OAS of (4). We solve the Black-Scholes formula for the implied volatility as the initial guess.

The fast convergence of the differential tree method is tabulated in Figure 11. The running time is quadratic: about $0.00000081 \times n^2$ seconds for the American call and $0.00000045 \times n^2$ seconds per iteration for the American put. Note that American calls will not be exercised early in our case of non-dividend-paying stock. Increasing the number of partitions may sometimes decrease the running time due to the reduction of the number of iterations. This is the case for the American put at $n = 600$.

6 Conclusions

This paper presented the differential tree method, then applied it to representative computational problems in asset pricing. The method was furthermore shown to be extremely efficient. In terms of software development, the method results in a program structure which almost parallels that of its dual, the pricing module; only the derivatives calculation and adjustments to the approximate root at the end of each iteration need to be added.

Other interesting applications are easy to think of. Take the calibration of interest rate models again. For finite-dimensional models, the limited number of parameters makes matching exactly the term structures impossible. For them, one may aim to minimize the mean square error $\| P(x) - p \|^2$ between the market and the model-implied term structures. By providing the derivatives efficiently, the differential tree method can easily solve this optimization problem. We conclude that, as a general computing paradigm, the differential tree method has potential for efficient calibration of any tree-based models.

Acknowledgments

The author thanks George Wei-Tso Chan, Wei-Jui Chen, and Chih-Jen Lin for assistance and discussions.

References

[1] Backus, D., S. Foresi, and S. Zin. "Arbitrage Opportunities in Arbitrage-Free Models of Bond Pricing." Manuscript, April 16, 1996. To appear in *Journal of Business and Economic Statistics*.

[2] Bjerksund, P., and G. Stensland. "Implementation of the Black-Derman-Toy Interest Rate Model." *Journal of Fixed Income*, September 1996, pp. 67–75.

[3] Black, F., E. Derman, and W. Toy. "A One-Factor Model of Interest Rates and Its Application to Treasury Bond Options." *Financial Analysts Journal*, January–February 1990, pp. 33–39.

[4] Canabarro, E. "Where Do One-Factor Interest Rate Models Fail?" *Journal of Fixed Income*, September 1995, pp. 31–52.

[5] Chen, W.-J. *Calibrating Interest Rate Models with Differential Tree Algorithms: the Case of Black-Derman-Toy Model.* Master's Thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, July 1997.

[6] Fabozzi, F. J. *Fixed Income Mathematics: Analytical & Statistical Techniques.* Revised ed. Chicago: Probus, 1991.

[7] Hildebrand, F. B. *Advanced Calculus for Applications.* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[8] Ho, T. S. Y., and S.-B. Lee. "Term Structure Movements and Pricing Interest Rate Contingent Claims." *Journal of Finance*, Vol. 41, 5 (December 1986), pp. 1011–1029.

[9] Hull, J. C. *Options, Futures, and Other Derivatives.* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[10] Hull, J. C., and A. White. "Pricing Interest-Rate-Derivative Securities." *Review of Financial Studies*, Vol. 3, 4 (1990), pp. 573–592.

[11] Ingersoll, J. E., Jr. "Interest Rates." In J. Eatwell, M. Milgate, and P. Newman (Ed.), *The New Palgrave: Finance*, New York: Norton, 1987.

[12] Jamshidian, F. "Forward Induction and Construction of Yield Curve Diffusion Models." *Journal of Fixed Income*, June 1991, pp. 62–74.

[13] Lyuu, Y.-D. *Introduction to Financial Computation: Principles, Mathematics, Algorithms.* Manuscript. Taipei, Taiwan: National Taiwan University, 1995.

[14] Lyuu, Y.-D. "Very Fast Algorithms for Barrier Option Pricing and the Ballot Problem." *Journal of Derivatives*, Spring 1998, pp. 68–79.

[15] Patterson, D. A., and J. H. Hennessy. *Computer Organization & Design: the Hardware/Software Interface.* San Mateo, CA: Morgan Kaufmann, 1994.

[16] Sandmann, K., and D. Sondermann. "A Note on the Stability of Lognormal Interest Rate Models and the Pricing of Eurodollar Futures." *Mathematical Finance*, Vol. 7, 2 (April 1994), pp. 119–125.

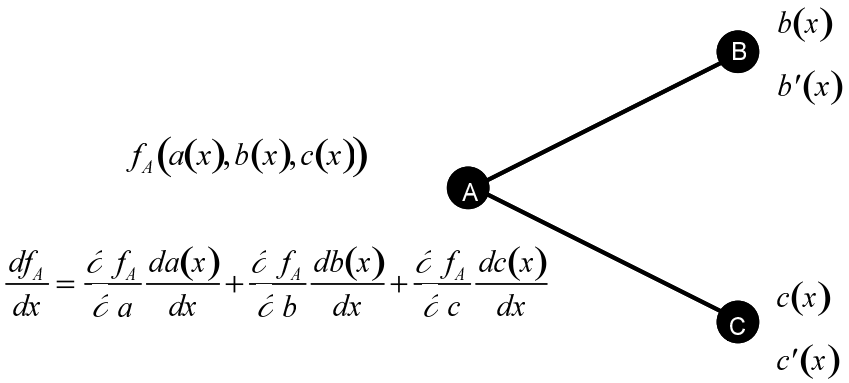

Figure 1: THE DIFFERENTIAL TREE METHOD. Computation at node A is driven by inputs from B and C.

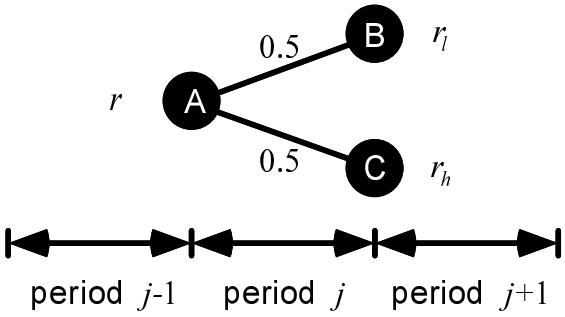

Figure 2: BINOMIAL INTEREST RATE PROCESS. From node A, there are two equally likely scenarios for the short rates. Rate r is applicable to node A for period j. Rate r_l is applicable to node B and rate r_h to node C, both for period $j+1$.

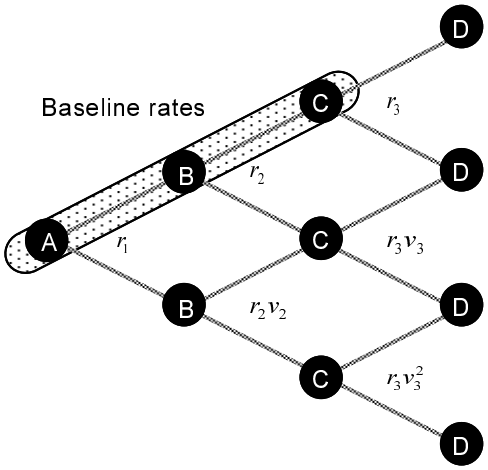

Figure 3: Binomial tree for the BDT model.

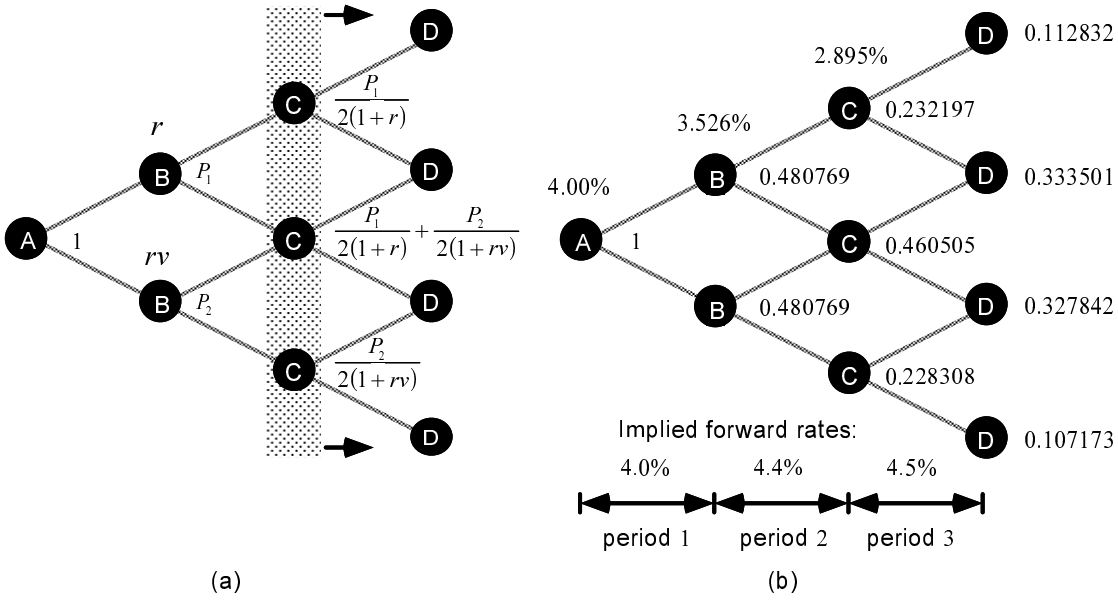

Figure 4: Sweeping a line across time forward to compute the binomial state price tree. The state price at a node is a weighted sum of the state prices of its two predecessors.

Number of Years	Average number of iterations	Sample deviation	Number of Years	Average number of iterations	Sample deviation
100	3.474747	0.519196	1100	2.926297	0.394491
200	3.236181	0.436405	1200	2.917431	0.394738
300	3.157192	0.373056	1300	2.923788	0.379878
400	3.085213	0.384873	1400	2.922802	0.373977
500	3.020040	0.414660	1500	2.893262	0.399234
600	2.973289	0.431584	1600	2.870544	0.415614
700	2.951359	0.428502	1700	2.847557	0.430938
800	2.929912	0.430495	1800	2.831573	0.439798
900	2.923248	0.421481	1900	2.817272	0.447092
1000	2.919920	0.412023	2000	2.806903	0.451480

Figure 5: CONVERGENCE OF THE DIFFERENTIAL TREE METHOD. The zero-coupon bond yield is described by $0.06 + 0.05 \times \ln t$. The time partition is one period per year. Sample deviation refers to that of the number of iterations. The termination condition is 10^{-13} relative error.

Number of Years	Backward induction	Forward induction	Number of Years	Backward induction	Forward induction
100	1.8	0.2	1100	2237.1	24.5
200	14.2	0.8	1200	2887.5	29.5
300	47.5	1.8	1300	3703.7	35.2
400	109.2	3.2	1400	4632.5	41.1
500	207.3	5.0	1500	5557.0	46.6
600	352.2	7.1	1600	6682.1	52.2
700	560.0	9.6	1700	8099.5	58.7
800	844.7	12.6	1800	9286.6	65.3
900	1223.5	15.9	1900	11088.6	72.9
1000	1690.7	20.1	2000	12745.2	80.9

Figure 6: BACKWARD INDUCTION VERSUS FORWARD INDUCTION. The setup is the same as Figure 5. All times are measured in seconds obtained on a 200MHz Sun ULTRA-SPARC with 256MB of DRAM. The termination condition is 10^{-13} relative error. Data are from Chen [1997].

Number of Years	Running time	Number of Years	Running time	Number of Years	Running time
3000	398.880	39000	8562.640	75000	26182.080
6000	1697.680	42000	9579.780	78000	28138.140
9000	2539.040	45000	10785.850	81000	30230.260
12000	2803.890	48000	11905.290	84000	32317.050
15000	3149.330	51000	13199.470	87000	34487.320
18000	3549.100	54000	14411.790	90000	36795.430
21000	3990.050	57000	15932.370	120000	63767.690
24000	4470.320	60000	17360.670	150000	98339.710
27000	5211.830	63000	19037.910	180000	140484.180
30000	5944.330	66000	20751.100	210000	190557.420
33000	6639.480	69000	22435.050	240000	249138.210
36000	7611.630	72000	24292.740	270000	313480.390

Figure 7: FORWARD INDUCTION (CONTINUED). We continue the benchmarking in Figure 6 for forward induction except that the termination condition is 10^{-6} relative error, and the times are measured on a slower 75MHz Sun SPARCstation 20 with 64MB of DRAM.

Number of partitions	Running time	Number of partitions	Running time
300	7.310	1500	166.130
600	26.400	1800	291.120
900	59.480	2100	497.130
1200	106.130	2400	diverge

Figure 8: FORWARD INDUCTION: THE 30-YEAR CASE. The setup is the same as Figure 7 except that $r^* = 0.08$, and the tree covers 30 years. The algorithm fails to terminate at 2,400.

Number of partitions	Running time	Number of partitions	Running time	Number of partitions	Running time
2100	242.110	4300	956.470	8500	3585.680
2300	291.850	4500	1037.540	9500	4500.570
2500	349.140	4700	1126.620	10500	5475.320
2700	407.370	4900	1202.000	11500	6579.760
2900	468.490	5100	1313.750	12500	7798.940
3100	527.100	5300	1394.370	13500	9191.240
3300	588.520	5500	1498.260	14500	10697.090
3500	651.360	5700	1609.560	15500	12167.020
3700	727.550	5900	1728.960	16500	13748.310
3900	804.130	6500	2067.470	17500	15732.760
4100	858.850	7500	2750.240	18500	17537.810

Figure 9: FORWARD INDUCTION: THE 10-YEAR CASE. The setup is the same as Figure 8 except that the tree covers 10 years. The algorithm fails to terminate at 19,000.

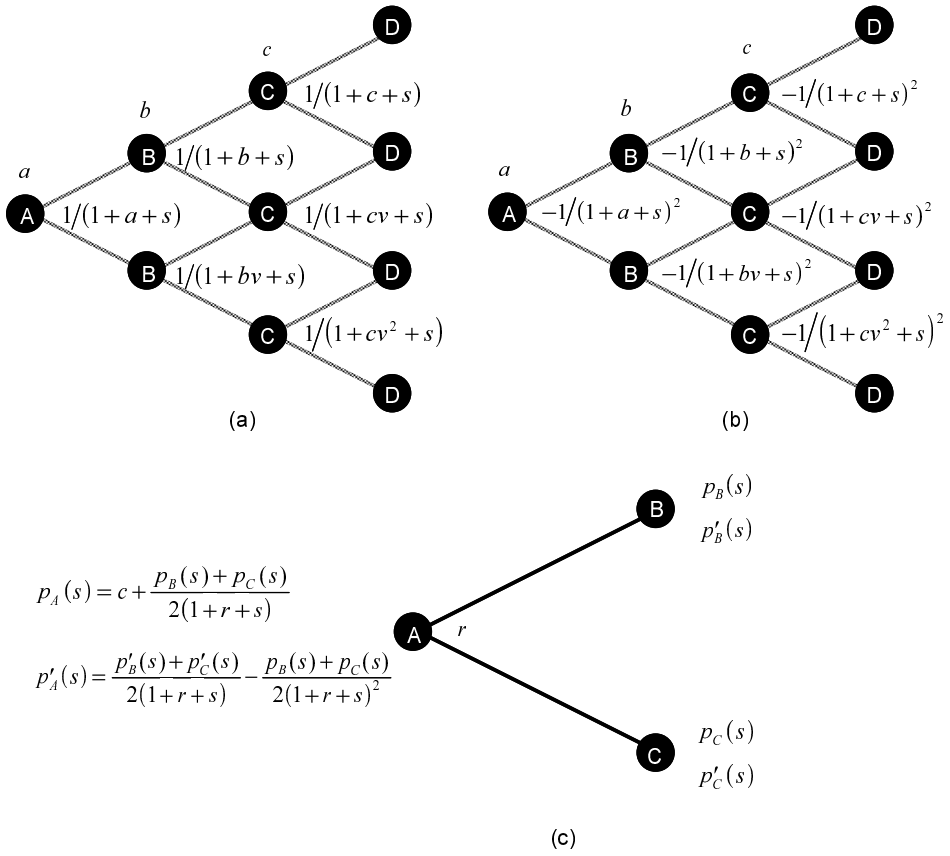

Figure 10: The differential tree for spread. (a) The original binomial interest rate tree with short rates replaced by the discount factors. (b) The derivatives of the numbers in (a). (c) The simultaneous evaluation of a function and its derivative with the binomial tree structure.

American call			American put		
Number of partitions	Running time	Number of iterations	Number of partitions	Running time	Number of iterations
100	0.008210	2	100	0.013845	3
200	0.033310	2	200	0.036335	3
300	0.072940	2	300	0.120455	3
400	0.129180	2	400	0.214100	3
500	0.201850	2	500	0.333950	3
600	0.290480	2	600	0.323260	2
700	0.394090	2	700	0.435720	2
800	0.522040	2	800	0.569605	2

Figure 11: IMPLIED VOLATILITY OF AMERICAN OPTIONS. The computer setup uses a PC equipped with an Intel 166MHz Pentium MMX and 32MB of DRAM, running Microsoft Windows 95. The program terminates when the volatility improves by less than 10^{-5}. The parameters for the American options are 49 (stock price), 50 (exercise price), 2.39 (option price), 5% (interest rate), and 140 (days to maturity). The implied volatility is found to be 19.95% for the call and 18.55% for the put.