

# **Binomial Trees for the Short Rate**

**Prasanth Karumanchi  
Ganesh Malligeswaran  
Andrew Vizcarra**

September 13, 2003

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Binomial Short Rate Trees for Black Derman Toy Model</b>	<b>4</b>
2.1	BDT Model Fitted to the Yield Curve only . . . . .	4
2.2	BDT Model fitted to the Yield Curve and Volatility . . . . .	5
2.3	Binomial Short Rate Trees for Ho-Lee . . . . .	6
<b>3</b>	<b>Pricing Bond Options</b>	<b>6</b>
3.1	Discount Bond Options . . . . .	6
3.2	European Swaptions . . . . .	7
<b>4</b>	<b>Data and Results</b>	<b>7</b>
4.1	Description of Data . . . . .	7
4.2	Results . . . . .	8
4.2.1	BDT Models vs. Strike Price . . . . .	8
4.2.2	BDT Models vs. Option Maturity . . . . .	9
4.2.3	Comparing the Two Variants of BDT . . . . .	10
4.2.4	BDT-Yield vs. Ho-Lee (Varying Volatility) . . . . .	11
4.2.5	Swaption vs. Volatility . . . . .	12
<b>5</b>	<b>Conclusions and Recommendations</b>	<b>14</b>
<b>6</b>	<b>Appendix</b>	<b>16</b>
6.1	BDT Model fitted to the Yield Curve Only . . . . .	16
6.1.1	Calculation of Short Rate Tree fitted to Yield Curve . . . . .	16
6.1.2	Newton-Ralphson Method for BDT fitted to Yield Curve . . . . .	16
6.1.3	Formulation of BDT Equation fitted to Yield Curve . . . . .	17
6.2	BDT model fitted to Yield Curve and Volatility . . . . .	18
6.2.1	Calculation of Short Rate Tree fitted to Yield Curve and Volatility . . . . .	18
6.2.2	Newton-Ralphson Method for Determining $P_u$ . . . . .	19
6.2.3	Formulation of $P_u$ Equation . . . . .	20
6.2.4	Newton-Ralphson Method for Calculating $U(\cdot)$ and $\sigma(\cdot)$ . . . . .	20
6.2.5	Formulation of $U(\cdot)$ and $\sigma(\cdot)$ Equation . . . . .	22
6.3	Ho Lee model fitted to Yield Curve . . . . .	22
6.3.1	Calculation of Short Rate Tree . . . . .	22
6.3.2	Newton-Raphson Method for the Ho-Lee Model . . . . .	23
6.3.3	Formulation of the Equation for Ho-Lee model . . . . .	24
6.4	Pricing Discount Bond Options . . . . .	25
6.4.1	BDT Bond Option Price fitted to Yield Only . . . . .	25
6.4.2	BDT Bond Option Price fitted to Yield and Volatility . . . . .	26
6.4.3	Ho-Lee Bond Option Price . . . . .	27
6.5	BDT model for Swaptions . . . . .	28
6.5.1	Swaption Pricing . . . . .	28
6.5.2	Validation of Coupon Dates . . . . .	29

## List of Figures

1	BDT-Yield-only vs. Strike Price . . . . .	8
2	BDT-Yield-Volatility vs. Strike Price . . . . .	9
3	BDT-Yield-only vs. Option Maturity . . . . .	9
4	BDT-Yield-Volatility vs. Option Maturity . . . . .	10
5	Difference of the Option Prices for the 2 BDT models . . . . .	10
6	BDT-Yield-Only vs. Volatility . . . . .	12
7	Ho-Lee vs. Volatility . . . . .	12
8	Swaption vs. Volatility (small sigma's) . . . . .	13
9	Swaption vs. Volatility (large sigma's) . . . . .	13

# 1 Introduction

The motivation of the construction of short rate trees for models consistent with the term structure is similar to that of the binomial trees for the underlying asset pricing. The main idea of the tree is to discretize the stochastic differential equation for the short rate to return the observed prices of pure discount bonds and their volatilities. These trees can also be used to price derivatives with the discount rate varying from node to node.

We divide the yield curve into  $i = 1, \dots, N$  equal segments, each having length  $\Delta t$ , and define the initial yield and volatility curves:

$P(i)$ : Price at time 0 of a pure discount bond maturing at time  $i\Delta t$ .  
 $R(i)$ : yield at time 0 on a pure discounting bond maturing at time  $i\Delta t$ .  
 $\sigma_R$ : volatility at time 0 of  $R(i)$ .

We look into two of the models implemented by building binomial trees - the simple additive Ho-Lee Model and the more popular Black-Derman-Toy (BDT) Model. Most of the material in this project had been obtained from the book : Implementing Derivative Models by Clewlow and Strickland (Clewlow and Strickland (1997)). A few internet resources have also been used which are referenced in the bibliography section.

## 2 Binomial Short Rate Trees for Black Derman Toy Model

The BDT model is a lognormal model and hence there is a compromise on the analytical flexibility (that is characteristic of normal models) in this model and hence we have to build the short rate trees until the end of the life of the underlying instrument. The risk-neutral probabilities of the of the binomial branches for this model are assumed to be equal to 1/2.

### 2.1 BDT Model Fitted to the Yield Curve only

By definition, the initial short rate  $r$  is the yield on the bond which matures at the end of first period  $\Delta t$ . In this model, we choose the short rates  $r_U$  and  $r_D$ , at nodes  $U$  and  $D$ , to match the initially specified curves. To match the initial yield curve we need the tree to correctly price a two-period bond. To match the initial volatility curve with the yield curve, the following equation needs to be satisfied:

$$volatility = \frac{1}{2} \ln \left( \frac{yield_{up}}{yield_{down}} \right)$$

where  $yield_{up}$  and  $yield_{down}$  are determined from the tree as the yields on the bonds maturing at time 2 as seen from the nodes  $U$  and  $D$ . So  $r_U$  and  $r_D$  are computed in order to satisfy these requirements. The other rates  $r_{.U}$  and  $r_{.D}$  are computed analogously.

To construct the tree efficiently, we use the technique of forward induction by Jamshedian(1991), which states that the level of short rate at time  $t$  in the BDT model is given by

$$r(t) = U(t)e^{\sigma(t)z(t)}$$

where  $U(t)$  is the median of the distribution for  $r$  at time  $t$ ,  $\sigma(t)$  is the level of short-rate volatility and  $z(t)$  is the level of Brownian motion. Assuming  $\sigma(t)$  is a constant, we only need to determine

$U(i)$  and  $\sigma(i)$ .

We index our tree using  $(i, j)$ , where  $i = 0, \dots, T$  denotes the time and  $j = -i, -i+1, \dots, i-1, i$  denotes the state. The level of the short rate in the tree is represented as

$$r_{i,j} = U(t)e^{\sigma(i)j\sqrt{\Delta t}}$$

Now, let us assume that  $Q_{i,j}$  is the value at time zero, of a security that pays \$1 if node  $(i, j)$  is reached and doesn't pay anything otherwise. The price of the pure discount bonds maturing at time  $(i+1)\Delta t$  can be expressed in terms of  $Q$  and a one-period discount factor at time  $i\Delta t$ .

$$P(i+1) = \sum_j Q_{i,j}d_{i,j}$$

where  $d_{i,j}$  denotes the price at the time  $i\Delta t$

$$d_{i,j} = \frac{1}{1 + r_{i,j}\Delta t}$$

Since the above equation is lognormal, we cannot obtain the value of  $U(i)$  explicitly and so we use a numerical method, namely the Newton-Ralphson method, to obtain this value. We then find  $Q_{i,j}$  according to the following equations

$$\begin{aligned} Q_{i,i} &= \frac{1}{2}Q_{i-1,i-1}d_{i-1,i-1} \\ Q_{i,j} &= \frac{1}{2}Q_{i-1,j-1}d_{i-1,j-1} + \frac{1}{2}Q_{i-1,j+1}d_{i-1,j+1} \quad j = -i+1 \text{ to } i-1 \\ Q_{i,-i} &= \frac{1}{2}Q_{i-1,-i+1}d_{i-1,-i+1} \end{aligned}$$

## 2.2 BDT Model fitted to the Yield Curve and Volatility

The steps for developing a BDT model to fit the yield curve and volatility are same as the ones followed in the model fitted to the yield curve only except that the volatility is not constant but is time dependent ( $\sigma(t)$ ).

Assume  $P_U(i)$  and  $P_D(i)$  to be the discount functions evaluated at the ‘‘up-step’’ node  $U$  and ‘‘down-step’’ node  $D$  respectively and  $R_U(i)$  and  $R_D(i)$  as the corresponding discount bond yields. With this notation, we then find that

$$P_D(i) = P_U(i)e^{-2\sigma R(i)\sqrt{\Delta t}}$$

where  $P_U(i)$  is the solution of the following equation:

$$P_U(i) + P_U(i)e^{-2\sigma R(i)\sqrt{\Delta t}} = 2P(i)(1 + r_{0,0}\Delta t)$$

We now define the state prices determined from the nodes  $U$  and  $D$  as follows:

- $Q_{U,i,j}$ : The value of a security that pays \$1 if node  $(i, j)$  is reached and 0 otherwise (at  $U$ ).
- $Q_{D,i,j}$ : The value of a security that pays \$1 if node  $(i, j)$  is reached and 0 otherwise (at  $D$ ).

We also have, by definition,  $Q_{U,1,1} = Q_{D,1,-1} = 1$ . Similar to the previous model, we obtain the following equations

$$\begin{aligned} P_U(i+1) &= \sum_j Q_{U,i,j} d_{i,j} \\ P_D(i+1) &= \sum_j Q_{D,i,j} d_{i,j} \end{aligned}$$

where

$$d_{i,j} = \frac{1}{1 + r_{i,j} \Delta t} = \frac{1}{1 + U(i) \exp(\sigma(i)j \sqrt{\Delta t} \Delta t)}$$

which we solve for  $U(i)$  and  $\sigma(i)$ .

### 2.3 Binomial Short Rate Trees for Ho-Lee

The Ho-Lee model is an additive model unlike the BDT model which is a multiplicative model. The short rate in the Ho Lee model is defined as

$$r_{t+1} = r_t + \mu_{t+1} + \varepsilon_{t+1}$$

where

$$\varepsilon_{t+1} = \begin{cases} +\sigma & \text{with probability } \frac{1}{2} \\ -\sigma & \text{with probability } \frac{1}{2} \end{cases}$$

$$q_u = q_d = \frac{1}{2} \frac{1}{1 + \frac{r}{2}}$$

The parameters in this model are chosen so as to facilitate the recombination of the branches. The choice of  $\sigma$  is arbitrary, but the values of  $\mu_t$  are so chosen such that the observed yield curve is generated.

The state prices are used in this model to eliminate the repetitive computation of the tree for the valuation of different bonds in a similar manner to the BDT manner. Using the state prices, one could use the same tree to value all fixed income securities.

## 3 Pricing Bond Options

### 3.1 Discount Bond Options

We know that if  $C_{i,j}$  represents the value of a contingent claim at node  $(i, j)$  then the value at this node is derived from the two nodes of time step  $(i+1)$  it is connected to, using the following equation:

$$C_{i,j} = \frac{1}{2} d_{i,j} [C_{i+1,j+1} + C_{i+1,j-1}]$$

We now price a call option maturing at time  $T$  on a discount bond maturing at time  $s$  ( $T \leq s$ ) with strike price  $K$ .

Let  $s = N_s \Delta t$  and  $T = N_T \Delta t$ . Let  $P_{s,i,j}$  represent the value of the  $s$ -maturity bond at node  $(i, j)$ . Hence the value of the  $s$ -maturity bond at every step is given by

$$P_{s,i,j} = \frac{1}{2} d_{i,j} [P_{s,i+1,j+1} + P_{s,i+1,j-1}] \quad \forall \text{ nodes } j \text{ at time step } i$$

Next, we evaluate the maturity condition for the option at all of the nodes for time step  $N_T$ :

$$C_{N_T,j} = \max\{0, P_{s,N_T,j} - K\} \quad \forall j \text{ at } N_T$$

For European options we can obtain the call price by the repeated application of the equation for  $C(i, j)$  from  $i = N_T - 1$  to 0. For the American Option we need to make this equation flexible for an early exit. The modified equation is given by:

$$C_{i,j} = \max\{P_{s_{i,j}} - K, \frac{1}{2}d_{i,j}[C_{i+1,j+1} + C_{i+1,j-1}]\}$$

### 3.2 European Swaptions

Recall that the swaption gives the holder the right to pay a fixed interest rate and receive floating interest rate, called payer swaption. It is equivalent to a put on a fixed rate bond with strike price equal to the principal of the swap, and the coupon rates equal to half the swap rate if the reset dates are half-yearly. If the swaption is a receiver swaption, that is it gives the holder the right to pay floating and receive fixed interest rate then it is equal to the call on a fixed rate bond. Let us assume that  $B_{i,j}$  represent the value of the fixed rate bond at node (i,j) in the tree and coupon/2 the cash flow at each coupon date.

We initialize the underlying of the swap, a fixed rate bond at every node at time steps  $N_s$  and  $B_{N_s,j}$ :

$$B_{N_s,j} = 1 + \frac{\text{coupon}}{2}$$

Taking discounted expectations until time  $T$ , we apply backward induction for coupon bond price:

$$B_{i,j} = \frac{1}{2}d_{i,j}(B_{i+1,j+1} + B_{i+1,j-1}) \quad \forall \text{ nodes } j \text{ at time step } i$$

We then use the state prices at all nodes at time step  $N_T$ , the payer and receiver swaption prices are calculated as

$$\begin{aligned} \text{payer swaption} &= \sum_j Q_{N_T,j} \max\{0, 1 - B_{N_s,j}\} \\ \text{receiver swaption} &= \sum_j Q_{N_T,j} \max\{0, B_{N_s,j} - 1\} \end{aligned}$$

## 4 Data and Results

### 4.1 Description of Data

The data used for the initial yield curve was that of the Euro Yield Curve. The curve includes maturities of one year up to 30 years. It is based on yield observations of actively traded bonds in Euro, weighted by the stock of bonds issued in Euro. Six sets of recent daily data were considered: starting from Tuesday, April 29, 2003 until Wednesday, May 7, 2003.

To obtain the initial volatility, which is needed in the BDT model, a 5-day volatility is adapted (5 “working days” a week). Specifically, the volatility used is the standard deviation of the yields over the week as a percentage of their arithmetic average.

For the sake of comparison, another set of yield curve is obtained. This time it is the US zero-coupon yield curve data with maturities of one year up to 5 years. A series of 30 monthly data was used.

## 4.2 Results

We ran several simulations on the bond option (European put) prices using the two variants of the BDT model and also with the Ho-Lee Model. We also do swaption pricing and observed its behavior versus other parameters.

All the results are shown in a graph and the analysis and observations are all based on these graphs.

### 4.2.1 BDT Models vs. Strike Price

In this simulation, we varied the values of the strike price. All other parameters are held constant:

$\sigma$	0.01	(used by BDT fitted to yield only)
$\Delta t$	1	
bond maturity	10 years	
option maturity	6 years	
initial yield curve	Euro yield curve	
initial volatility	Euro yield curve volatility	(used by BDT-Yield-Volatility)

Figure 1: BDT-Yield-only vs. Strike Price

As can be expected, Figures 1 and 2 depict the put option payoff function although there seems to be some random error which makes the graph not straight.

Comparing Figures 1 and 2, it can be observed that there is little, if any, difference in the option prices.

Figure 2: BDT-Yield-Volatility vs. Strike Price

#### 4.2.2 BDT Models vs. Option Maturity

The following are the values used:

$\sigma$	0.01	(used by BDT fitted to yield only)
$\Delta t$	1	
bond maturity	10 years	
strike price	1	
initial yield curve	Euro yield curve	
initial volatility	Euro yield curve volatility	(used by BDT-Yield-Volatility)

while the option maturity is varied.

Figure 3: BDT-Yield-only vs. Option Maturity

Figure 4: BDT-Yield-Volatility vs. Option Maturity

Both graphs given in Figures 3 and 4 depict a decreasing linear function. This is reasonable since as the option maturity nears the bond maturity, one has a lesser degree of freedom in terms of the time horizon of the life of the bond.

Again, there seems to be no difference between the two graphs.

#### **4.2.3 Comparing the Two Variants of BDT**

Figure 5: Difference of the Option Prices for the 2 BDT models

Since there seems to be no difference between the BDT model fitted to yield only and the BDT model fitted to yield and volatility, we use another data set to determine if this is really true. This is extremely important since this might imply that the additional burden of fitting the volatility is not worth it.

Here, we used the US zero-coupon yield curve as the initial yield curve. The other parameters used were:

initial volatility	US zero-coupon yield curve volatility	(used by BDT-Yield-Volatility)
$\sigma$	first entry in the initial volatility	(used by BDT fitted to yield only)
$\Delta t$	1	
bond maturity	3 years	
option maturity	2 years	
strike price	1	
initial yield curve	US zero-coupon yield curve	

We computed for the difference between the prices of the 2 BDT models for a span of 29 monthly data.

Figure 5 clearly shows a significant difference among the prices computed between the two models. Upon inspection, we realized that the volatility from the Euro yield curve data were not significantly different over several observations. Moreover, the value was pretty close to the constant value we used in the BDT model fitted to the yield curve only.

#### 4.2.4 BDT-Yield vs. Ho-Lee (Varying Volatility)

We now try to compare the BDT-Yield-only model with that of the Ho-Lee model. Note that the Ho-Lee model is also fitted to the same initial yield curve.

$\Delta t$	1
bond maturity	3 years
option maturity	2 years
strike price	1
initial yield curve	Euro yield curve

Surprisingly, Figures 6 and 7 have opposing results. If we look at the order of magnitude, Figure 5 really looks like a constant when compared to the prices for that of Ho-Lee which seems to be an increasing function. The behavior of the Ho-Lee model may be attributed to its being an additive model. This is one of the drawbacks of the Ho-Lee model: if we increase the time horizon, we will get extreme values, very large interest rates or very low and even negative interest rates.

Figure 6: BDT-Yield-Only vs. Volatility

Figure 7: Ho-Lee vs. Volatility

#### 4.2.5 Swaption vs. Volatility

Lastly, we study the behavior of the swaption against volatility. Figure 8 suggest a decreasing function. However, this is deceiving since here the option prices are almost constant separated only by a very small number (look at the scale on the prices). This only means that small volatility will have little or no effect at all on the option price.

Figure 9 is the same plot but using larger values of sigma. We clearly see an increasing trend

here which is what should be expected.

Figure 8: Swaption vs. Volatility (small sigma's)

Figure 9: Swaption vs. Volatility (large sigma's)

## 5 Conclusions and Recommendations

This paper tried to study and compare bond option prices under the BDT and Ho-Lee model. As the literature suggests, we see a great inclination on the use of the BDT model as compared to the Ho-Lee model. Moreover, we also notice that fitting the volatility data along with the term structure might give acceptable values. And it is also comforting that that the expected behavior of these option prices, as we learned in class, was confirmed.

In terms of accuracy and popularity, our readings point to the use of other models like the Black-Karasinski model which uses trinomial trees. It would have been interesting if we could have compared the prices generated between this model and the BDT model fitted to both yield and volatility.

Another area of improvement for the BDT model fitted to the yield curve and volatility is in the computation of the initial volatility. There are several ways of computing for the volatility. One that is becoming popular especially in terms of financial time series is the GARCH(1,1) (Generalized Autoregressive Conditional Heteroskedasticity) model which is now widely used in estimating volatility.

In the quest for better accuracy comparisons, we could have compared the generated bond option prices with the actual bid and ask prices of the same option (this could be done by averaging the bid and ask prices and taking the difference of this average with the computed bond option price). It is actually in this direction that we tried to devote most of our time. However, we were not successful in obtaining actual bid and ask prices: either we found no actual data for a particular option or the data was insufficient. This is very unfortunate owing to the fact that there were a lot of databases available in the Krannert Library and even on the Internet; but we just couldn't (or don't know how to) find the data.

## References

- [1] Clewlow, L., Strickland, C. *Implementing Derivatives Models*: John Wiley, 1998, ISBN 0-471-96651-7
- [2] Chourdakis, K. (2003) Models for the short rate. *The Derivatives-on-Line Pages* [Online]. Available: <http://www.qmw.ac.uk/~te9001/DOL/DOLnode45.htm> [Accessed 05 May 2003]
- [3] Oeticker, T, Partl, H., Hyna, I. and Schlegl, E. *The (Not So) Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>* [Online] Available:<http://www.ctan.org/tex-archive/info/lshort> [Accessed 05 May 2003]

## 6 Appendix

The following are the Matlab codes used in this paper. Due to space restrictions, we did not include here the simulation codes.

### 6.1 BDT Model fitted to the Yield Curve Only

#### 6.1.1 Calculation of Short Rate Tree fitted to Yield Curve

```
function [r,d,Q]=BDTYield(sigma,N,T,initial_y);
%computes for r, d and Q

for i=1:N
    R(i)=initial_y(i);
    P(i)=1/(1+R(i)*dt)^(i*dt);
end

%initialize the first node
Q(1,1)=1; U(1)=R(1); r(1,1)=R(1); d(1,1)=1/(1+r(1,1)*dt);

%evolve tree for the short rate
for i=2:N

    %update pure security prices at i
    Q(i,1)=0.5*Q(i-1,1)*d(i-1,1);
    Q(i,i)=0.5*Q(i-1,i-1)*d(i-1,i-1);
    if i > 2
        for j=2:i-1
            Q(i,j)=0.5*Q(i-1,j-1)*d(i-1,j-1)+0.5*Q(i-1,j)*d(i-1,j);
        end
    end

    %use the Newton-Raphson method to solve for U(i)
    U(i)=newton_bdt_y('bdteqn_y',U(i-1),i,Q(i,:),P(i),sigma,dt);

    %set r(.) and d(.)
    for k=1:i
        j(k)=i-1-2*(k-1);
        r(i,k)=U(i)*exp(sigma*j(k)*sqrt(dt));
        d(i,k)=1/(1+r(i,k)*dt);
    end
end
```

#### 6.1.2 Newton-Ralphson Method for BDT fitted to Yield Curve

```
function [x,F]=newton_bdt_y(FunFcn,x0,i,Q,P,sigma,dt,nmax,tol,trace)
%computes for the approximate root and the error

%check sufficiency of parameters
```

```

if nargin<7, help full_new, return, end

%set default not to print values
if nargin<10
    trace=0;
end;

%set error tolerance
if nargin<9
    tol=0.00001;
end;

%set maximum number of iterations
if nargin<8
    nmax=50;
end;

f=[];J=[];x=[x0'];F=[ ]; x1=x0;

%start iteration
for n=2:nmax

    %evaluate function and its derivative at x1
    [f,J]=feval(FunFcn,x1,i,Q,P,sigma,dt);

    % stop failures
    if det(J)==0|det(J)==Inf,x=Inf;F=Inf;return;end

    %set next estimate
    x1=x1-f/J;

    %determine if tolerance is met;
    %if number of iterations already reached the maximum;
    %and whether to print values
    if norm(f)<tol
        n=nmax;
    elseif trace==0
        x=x1;F=f;
    else
        x=[x; x1'];F=[F; f'];
    end
end
end

```

### 6.1.3 Formulation of BDT Equation fitted to Yield Curve

```

function [q,J] = BDTeqn_y(p,i,Q,P,sigma,dt)
%evaluates the function values and derivative values at the current guess

```

```

%set initial values
x = p(1); q(1)=-P; J(1)=0;

for k=1:i

    %adjust j to go from -i to i
    j(k)=i-1-2*(k-1);

    %compute for constants in the equation
    ex=exp(sigma*j(k)*sqrt(dt))*dt;
    den=1+x*ex;

    %form equation and its derivative evaluated at the initial value
    q(1)=q(1)+Q(k)/den;
    J(1)=J(1)-Q(k)*ex/den^2;
end

```

## 6.2 BDT model fitted to Yield Curve and Volatility

### 6.2.1 Calculation of Short Rate Tree fitted to Yield Curve and Volatility

```

function [r,d,Qu,Qd]=BDTYieldVol(N,T,initial_y,initial_v);
%computes for r,d, Qu and Qd

%precompute constants
dt=T/N; sdt = sqrt(dt);

%initialize yield curve
for i=1:N
    R(i)=initial_y(i);
    P(i)=1/(1+R(i)*dt)^(i*dt);
    sigR(i)=initial_v(i);
end

%compute for Pu and Pd
for i=2:N
    %solve for Pu(i) using the Newton-Raphson method
    Pu(i)=newton_pu('pu_eqn',P(i),R(1),P(i),sigR(i),dt);

    %set Pd(i)
    Pd(i)=Pu(i)^(exp(-2*sigR(i)*sdt));
end

%initialize the first node
U(1)=R(1); r(1,1)=R(1); d(1,1)=1/(1+r(1,1)*dt); sig(1)=sigR(1);
Qu(1,1)=1; Qd(1,1)=1;

%evolve tree for the short rate

```

```

for i=2:N

    %use the Newton-Raphson method to solve for U(i) and sig(i)
    x=newton_u_sig('u_sig_eqn',[U(i-1),

sig(i-1)],i,Qu(i-1,:),Qd(i-1,:),[Pu(i),Pd(i)],dt);
    U(i)=x(1);
    sig(i)=x(2);

    %set r(.) and d(.)
    for k=1:i
        j(k)=i-1-2*(k-1);
        r(i,k)=U(i)*exp(sig(i)*j(k)*sdt);
        d(i,k)=1/(1+r(i,k)*dt);
    end

    %update pure security prices at i
    Qu(i,1)=0.5*Qu(i-1,1)*d(i,1);
    Qu(i,i)=0.5*Qu(i-1,i-1)*d(i,i-1);
    Qd(i,1)=0.5*Qd(i-1,1)*d(i,2);
    Qd(i,i)=0.5*Qd(i-1,i-1)*d(i,i);

    if i > 2
        for j=2:i-1
            Qu(i,j)=0.5*Qu(i-1,j-1)*d(i,j-1)+0.5*Qu(i-1,j)*d(i,j);
            Qd(i,j)=0.5*Qd(i-1,j-1)*d(i,j)+0.5*Qd(i-1,j)*d(i,j+1);
        end
    end
end
end

```

### 6.2.2 Newton-Ralphson Method for Determining $P_u$

```

function [x,F]=newton_pu(FunFcn,x0,r,P,sig,dt,nmax,tol,trace)
%computes for the approximate root and the error

%check sufficiency of parameters
if nargin<6, help full_new, return, end

%set default not to print values
if nargin<9
    trace=0;
end;

%set error tolerance
if nargin<8
    tol=0.00001;
end;

```

```

%set maximum number of iterations
if nargin<7
    nmax=50;
end;

f=[];J=[];x=[x0'];F=[ ]; x1=x0;

%start iteration
for n=2:nmax

    %evaluate function and its derivative at x1
    [f,J]=feval(FunFcn,x1,r,P,sig,dt);

    %stop failures
    if det(J)==0|det(J)==Inf,x=Inf;F=Inf;return;end

    %set next estimate
    x1=x1-f/J;

    %determine if tolerance is met;
    %if number of iterations already reached the maximum;
    %and whether to print values
    if norm(f)<tol
        n=nmax;
    elseif trace==0
        x=x1;F=f;
    else
        x=[x; x1'];F=[F; f'];
    end
end
end

```

### 6.2.3 Formulation of $P_u$ Equation

```

function [q,J] = pu_eqn(p,r,P,sig,dt)
%computes for the approximate root and the error

%set initial values
x = p(1);

%compute for constants in the equation
sdt=sqrt(dt); ex=exp(-2*sig*sdt);

%form equation and its derivative evaluated at the initial value
q(1)=x+x^ex-2*P*(1+r*dt); J(1)=1+ex*(x^(ex-1));

```

### 6.2.4 Newton-Ralphson Method for Calculating $U(\cdot)$ and $\sigma(\cdot)$

```

function [x,F]=newton_u_sig(FunFcn,x0,i,Qu,Qd,P,dt,nmax,tol,trace)

```

```

%check sufficiency of parameters
if nargin<7, help full_new, return, end

%set default not to print values
if nargin<10
    trace=0;
end;

%set error tolerance
if nargin<9
    tol=0.00001;
end;

%set maximum number of iterations
if nargin<8
    nmax=50;
end;

f=[];J=[];x=[x0'];F=[]; x1=x0;

%start iteration
for n=2:nmax

    %evaluate function and its derivative at x1
    [f,J]=feval(FunFcn,x1,i,Qu,Qd,P,dt);

    %stop failures
    if det(J)==0|det(J)==Inf,x=Inf;F=Inf;return;end

    %set next estimate
    x1=x1-f*transpose(inv(J));

    %determine if tolerance is met;
    %if number of iterations already reached the maximum;
    %and whether to print values
    if norm(f)<tol
        n=nmax;
    elseif trace==0
        x=x1;F=f;
    else
        x=[x; x1'];F=[F; f'];
    end
end
end

```

## 6.2.5 Formulation of $U(\cdot)$ and $\sigma(\cdot)$ Equation

```
function [q,J] = u_sig_eqn(p,i,Qu,Qd,P,dt)

%set initial values
x = p(1); y = p(2); q(1)=-P(1); q(2)=-P(2); J(1,1)=0; J(1,2)=0;
J(2,1)=0; J(2,2)=0;

for k=1:i-1

    %adjust j for Qu and Qd
    ju(k)=i-1-2*(k-1);
    jd(k)=ju(k)-2;

    %compute quantities in the equation
    exu=exp(y*ju(k)*sqrt(dt))*dt;
    denu=1+x*exu;

    exd=exp(y*jd(k)*sqrt(dt))*dt;
    dend=1+x*exd;

    %form equation and its derivative evaluated at the initial value
    q(1)=q(1)+Qu(k)/denu;
    q(2)=q(2)+Qd(k)/dend;

    J(1,1)=J(1,1)-Qu(k)*exu/denu^2;
    J(1,2)=J(1,2)-Qu(k)*x*exu*ju(k)*sqrt(dt)/denu^2;
    J(2,1)=J(2,1)-Qd(k)*exd/dend^2;
    J(2,2)=J(2,2)-Qd(k)*x*exd*jd(k)*sqrt(dt)/dend^2;
end
```

## 6.3 Ho Lee model fitted to Yield Curve

### 6.3.1 Calculation of Short Rate Tree

```
function [r,d,Q,P]=HoLeeYield(sigma,N,T,initial_y);
%computes for r, d, Q and P

%precompute constants
dt=T/N;

%initialize yield curve
for i=1:N
    R(i)=initial_y(i);
    P(i)=1/(1+R(i)*dt)^(i*dt);
end

%initialize the first node
Q(1,1)=1; u(1)=0; r(1,1)=R(1); d(1,1)=1/(1+r(1,1)*dt);
```

```

%evolve tree for the short rate
for i=2:N

    %update pure security prices at i
    Q(i,1)=0.5*Q(i-1,1)*d(i-1,1);
    Q(i,i)=0.5*Q(i-1,i-1)*d(i-1,i-1);
    if i > 2
        for j=2:i-1
            Q(i,j)=0.5*Q(i-1,j-1)*d(i-1,j-1)+0.5*Q(i-1,j)*d(i-1,j);
        end
    end
end

%use the Newton-Raphson method to solve for u(i)
u(i)=HLnewton('HLequation',r(1,1),i,Q(i,:),P(i),r(1,1),u,sigma,dt);

%set r(.) and d(.)
for k=1:i
    j(k)=i-1-2*(k-1);
    r(i,k)=r(1,1)+sigma*j(k)+sum(u);
    d(i,k)=1/(1+r(i,k)*dt);
end
end

```

### 6.3.2 Newton-Raphson Method for the Ho-Lee Model

```

function [x,F]=HLnewton(FunFcn,x0,i,Q,P,r,u,sigma,dt,nmax,tol,trace)
%computes for the approximate solution and error

%check sufficiency of parameters
if nargin<9, help full_new, return, end

%set default not to print values
if nargin<12
    trace=0;
end;

%set error tolerance
if nargin<11
    tol=0.00001;
end;

%set maximum number of iterations
if nargin<10
    nmax=50;
end;

f=[];J=[];x=[x0'];F=[ ]; x1=x0;

```

```

%start iteration
for n=2:nmax

    %evaluate function and its derivative at the x1
    [f,J]=feval(FunFcn,x1,i,Q,P,r,u,sigma,dt);

    %stop failures
    if det(J)==0|det(J)==Inf,x=Inf;F=Inf;return;end

    %set next estimate
    x1=x1-f/J;

    %determine if tolerance is met;
    %if number of iterations already reached the maximum;
    %and whether to print values
    if norm(f)<tol
        n=nmax;
    elseif trace==0
        x=x1;F=f;
    else
        x=[x; x1'];F=[F; f'];
    end
end
end

```

### 6.3.3 Formulation of the Equation for Ho-Lee model

```

function [q,J] = HLequation(p,i,Q,P,r_i,u,sigma,dt
%evaluates functions values including its derivatives

%set initial values
x = p(1); q(1)=-P; J(1)=0;

%get the sum of the previous values of u and store it to v
v=0; if i>2
    for n=2:i-1
        v=v+u(n);
    end
end

for k=1:i

    %adjust j to go from -i to i
    j(k)=i-1-2*(k-1);

    %compute constants in the equation
    r=(r_i+sigma*j(k)+v+x)*dt;

```

```

den=1+r;

%form equation and its derivative evaluated at the initial value
q(1)=q(1)+Q(k)/den;
J(1)=J(1)-Q(k)*dt/den^2;
end

```

## 6.4 Pricing Discount Bond Options

### 6.4.1 BDT Bond Option Price fitted to Yield Only

```

function [E,A]=BDTbond_option(sigma,Ns,NT,T,K,initial_y)
%computes for the American and European put prices

%precompute constants
dt=T/NT; s=N*s*dt;

%build short rate tree
[r,d,Q]=BDTYield(sigma,Ns,s,initial_y);

%adjust indices to start at 1 and end at N.+1
NT=NT+1; Ns=Ns+1;

%initialize maturity condition for pure discount bond
%underlying the option
for j=1:Ns
    Ps(Ns,j)=1;
end

%backward induction for pure discount bond price
for i=Ns-1:-1:1
    for j=1:i
        Ps(i,j)=d(i,j)*0.5*(Ps(i+1,j)+Ps(i+1,j+1));
    end
end

%initialize maturity condition for option
for j=1:NT
    C(NT,j)=max(0,K-Ps(NT,j));
end

%evaluate European option price using state prices
C(1,1)=0; for j=1:NT
    C(1,1)=C(1,1)+Q(NT,j)*C(NT,j);
end

E=C(1,1);

```

```

%backward induction for American put option price
for i=NT-1:-1:1
    for j=1:i
        C(i,j)=d(i,j)*0.5*(C(i+1,j)+C(i+1,j+1));
        C(i,j)=max(C(i,j),K-Ps(i,j));
    end
end

A=C(1,1);

```

#### 6.4.2 BDT Bond Option Price fitted to Yield and Volatility

```

function [E,A]=BDTbond_option(Ns,NT,T,K,initial_y,initial_v)
%computes for the American and European put prices

%precompute constants
dt=T/NT; s=Ns*dt;

%build short rate tree
[r,d,Qu,Qd]=BDTYieldVol(Ns,s,initial_y,initial_v);

%adjust indices to start at 1 and end at N.+1
NT=NT+1; Ns=Ns+1;

%initialize maturity condition for pure discount bond
%underlying the option
for j=1:Ns
    Ps(Ns,j)=1;
end

%backward induction for pure discount bond price
for i=Ns-1:-1:1
    for j=1:i
        Ps(i,j)=d(i,j)*0.5*(Ps(i+1,j)+Ps(i+1,j+1));
    end
end

%initialize maturity condition for option
for j=1:NT
    C(NT,j)=max(0,K-Ps(NT,j));
end

EC=C;

%backward induction for European put option price
for i=NT-1:-1:1
    for j=1:i

```

```

        EC(i,j)=d(i,j)*0.5*(EC(i+1,j)+EC(i+1,j+1));
    end
end

```

```

E=EC(1,1);

```

```

%backward induction for American put option price

```

```

for i=NT-1:-1:1
    for j=1:i
        C(i,j)=d(i,j)*0.5*(C(i+1,j)+C(i+1,j+1));
        C(i,j)=max(C(i,j),K-Ps(i,j));
    end
end

```

```

A=C(1,1);

```

### 6.4.3 Ho-Lee Bond Option Price

```

function [E,A]=HLbond_option(sigma,Ns,NT,T,K,initial_y)

```

```

%precompute constants

```

```

dt=T/NT; s=N*dt;

```

```

%build short rate tree

```

```

[r,d,Q,P]=HoLeeYield(sigma,Ns,s,initial_y);

```

```

%adjust indices to start at 1 and end at N*+1

```

```

NT=NT+1; Ns=N+1;

```

```

%initialize maturity condition for pure discount bond

```

```

%underlying the option

```

```

for j=1:Ns
    Ps(Ns,j)=1;
end

```

```

%backward induction for pure discount bond price

```

```

for i=N-1:-1:1
    for j=1:i
        Ps(i,j)=d(i,j)*0.5*(Ps(i+1,j)+Ps(i+1,j+1));
    end
end

```

```

%initialize maturity condition for option

```

```

for j=1:NT
    C(NT,j)=max(0,K-Ps(NT,j));
end

```

```

%evaluate European option price using state prices

```

```

C(1,1)=0; for j=1:NT
    C(1,1)=C(1,1)+Q(NT,j)*C(NT,j);
end

E=C(1,1);

%backward induction for American call option price
for i=NT-1:-1:1
    for j=1:i
        C(i,j)=d(i,j)*0.5*(C(i+1,j)+C(i+1,j+1));
        C(i,j)=max(C(i,j),K-Ps(i,j));
    end
end

A=C(1,1);

```

## 6.5 BDT model for Swaptions

### 6.5.1 Swaption Pricing

```

function C=swaption(sigma,N,NT,T,coupon,cdates,initial_y)
%computes for the swaption price

%cdates is in terms of dt

%precompute constants
dt=T/NT; Ns=N+NT; s=Ns*dt;

%build short rate tree
[r,d,Q]=BDTYield(sigma,Ns,s,initial_y);

%adjust indices to start at 1 and end at N*+1
NT=NT+1; Ns=Ns+1;

%initialize maturity condition for fixed side of swap
for j=1:Ns
    B(Ns,j)=1+coupon/2;
end

%derive the coupon bond_price in the tree via discounted expectation
for i=Ns-1:-1:NT
    for j=1:i
        if if_coupon_payment_date(i-NT,cdates)==1
            B(i,j)=d(i,j)*0.5*(B(i+1,j)+B(i+1,j+1)+coupon/2);
        else
            B(i,j)=d(i,j)*0.5*(B(i+1,j)+B(i+1,j+1));
        end
    end
end

```

```

end

%utilize the pure security prices for European swaption value
C=0;

for j=1:NT
    C=C+Q(NT,j)*max(0,1-B(NT,j));
end

```

### 6.5.2 Validation of Coupon Dates

```

function x=if_coupon_payment_date(n,cdates);

%default value of 0 means it is not a coupon payment date
x=0;

%get the length of the vector with the coupon dates.
len=length(cdates);

%set x=1 if n is the same as one of the coupon dates
for i=1:len
    if n==cdates(i)
        x=1;
        return;
    end
end
end

```

## References

- [1] Clewlow, L., Strickland, C. *Implementing Derivatives Models*: John Wiley, 1998, ISBN 0-471-96651-7
- [2] Chourdakis, K. (2003) Models for the short rate. *The Derivatives-on-Line Pages* [Online]. Available: <http://www.qmw.ac.uk/~te9001/DOL/DOLnode45.htm> [Accessed 05 May 2003]
- [3] Holden, Craig W., US Yield Curve Dynamics. *Modeling - How To Build Financial Models in Excel Spreadsheet* [Online]. Available: <http://www.spreadsheetmodeling.com> [Accessed 05 May 2003]
- [4] Euro par Yield Curve, Eurostat. [Online]. Available: <http://www.europa.eu.int/comm/eurostat/Public/datashop/print-product/EN?catalogue=Eurostat&product=eyc-EN-EN&mode-download> [Accessed 05 May 2003]