CHAPTER 8

The Black-Derman-Toy Model

8.1. Term Structure Model and Black-Derman-Toy Tree

In analogy to stock options, interest rate derivatives depend on their underlying that is generally the interest rate. The behavior of interest rates for all maturity terms will thus play a crucial role in the corresponding pricing scheme. In a one-factor term structure model, there is only one underlying process known as the *instantaneous short rate* r_t that defines the interest rate at any time t for a very short borrowing term. In this model, the time evolution of the entire yield curve will solely be driven by the behavior of this short-term interest rate. Derivatives can be priced in accordance with the short-rate process under risk-neutral preference as

$$f_0 = \hat{E}(e^{-\int_0^T r_t\, dt} f_T \mid r_0) \tag{8.1}$$

with discount factor that cumulates over generated interest rate path with current value r_0. The option payoff f_T in (8.1) is evaluated according to the realized rate at its maturity. Modern approach to the modeling of stochastic short rate started with the early equilibrium model by Vasicek[1] to the rigorous no-arbitrage formulation by Heath, Jarrow, and Morton[2]. For numerical pricing of options, however, it is sufficient to have a discrete tree-type model for risk-neutral short rate that exactly matches current yield curve and its stochastic properties. In this connection, Black, Derman and Toy[3] (BDT) have developed a simple binomial model that can be calibrated to fit the current term structures of zero-coupon bond prices and volatilities.

The BDT model has adopted a recombining binomial lattice for the risk-neutral short rate with discrete time interval of Δt as shown in Figure 8.1(a).

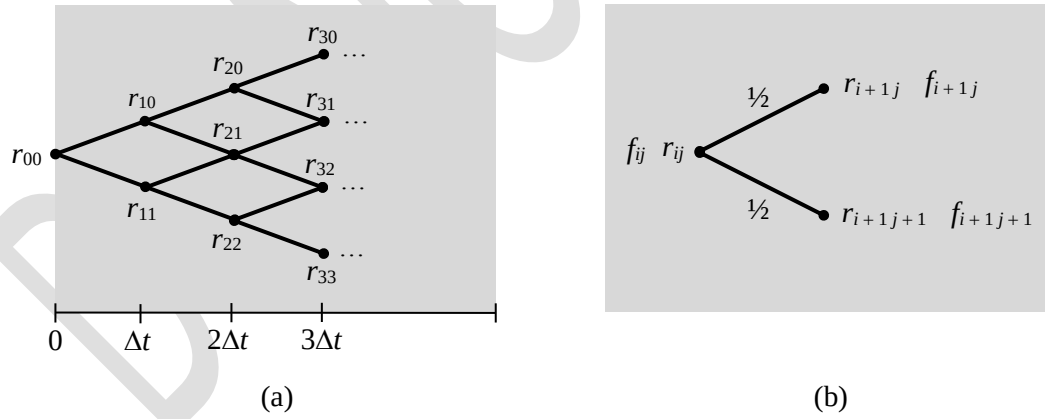


Figure 8.1: (a) BDT binomial model with recombining tree nodes. (b) Risk-neutral pricing of option along BDT subtree.

[1] See O.A. Vasicek, "An equilibrium characterization of the term structure", *Journal of Financial Economics*, Vol. **5**, Issue 2 (1977) p177-188.

[2] See D. Heath, R. Jarrow, and A. Morton, "Bond pricing and the term structure of interest rates; a discrete time approximation", *Journal of Financial and Quantitative Analysis*, Vol. **25**, No. 4 (1990) p419-440.

[3] See F. Black, E. Derman, and W. Toy, "A one-factor model of interest rates and its application to treasury bond options", *Financial Analysts Journal*, Vol. **46**, Issue 1 (1990) p33-39.

At time $t = i\Delta t$, there are $i + 1$ nodes from top to bottom of the tree with short rates defined to be r_{ij} where j runs from 0 to i. They represent the annualized one-period interest rates for the shortest borrowing term of Δt from time t to $t + \Delta t$. The BDT tree follows the Jarrow and Rudd parameterization with symmetric branching probabilities of $p = ½$. In general, the up and down factors will depend on time and the corresponding value of short rate. If we assume a non-stochastic structure for the short-rate volatility, it can be shown that its ratio will only depend on time and the entire column of short rates at time step i can be parameterized by two factors as[4]

$$r_{ij} = \alpha_i \, (\beta_i)^j \tag{8.2}$$

BDT tree provides a tool for evaluating the risk-neutral pricing of interest rate options as given by (8.1). Consider the one-step subtree as depicted in Figure 8.1(b), the risk-neutral pricing in (8.1) can be written as

$$f_{ij} = e^{-r_{ij}\Delta t} \, (\, ½ \, f_{i+1\,j} + ½ \, f_{i+1\,j+1} \,) \tag{8.3}$$

where $e^{-r_{ij}\Delta t}$ is the discount factor that cumulates over the one-step interest rate path with realized rate r_{ij}. The current price of the option f_{00} can be determined by iterating (8.3) backward in time starting from its maturity payoff $\psi(r_T)$ and along a tree with longer time horizon.

For short-rate tree with time interval Δt and horizon $T_{tree} = N_{tree}\Delta t$, it can be constructed by calibrating with the current term structures of zero-coupon bond prices $\{ P_0(\tau_1), P_0(\tau_2), \ldots , P_0(\tau_{N_{tree}+1}) \}$ and their volatilities $\{ \sigma_0(\tau_2), \ldots , \sigma_0(\tau_{N_{tree}+1}) \}$. The maturity terms of the bonds must coincide with the time structure of the tree for which $\tau_m = m\Delta t$. The size of Δt must be kept very small so as to improve accuracy and to cope with intermediate boundary condition for the option. In this respect, the current term structures with arbitrary time interval can be constructed through cubic spline interpolation as discussed in earlier chapter. Practically, the size of Δt can be taken as the shortest maturity term available on market bond data in order to maximize precision. For zero-coupon bond that matures at time τ_m, the forward bond price $P_{ij}(\tau_m)$ at previous tree node (i, j) should satisfy the risk-neutral pricing in (8.3) for subtree as

$$P_{ij}(\tau_m) = e^{-r_{ij}\Delta t} \, [\, ½ \, P_{i+1\,j}(\tau_m) + ½ \, P_{i+1\,j+1}(\tau_m) \,] \tag{8.4}$$

The current bond price and its volatility can be determined by iterating (8.4) backward in time starting from its maturity at time τ_m with $P_{mj}(\tau_m) = \$1$ for the entire column of tree nodes. The short-rate tree should generate the values consistent with the market term structures such that[5]

$$P_0(\tau_m) = P_{00}(\tau_m) \tag{8.5}$$

$$\sigma_0(\tau_m) = \frac{1}{2\sqrt{\Delta t}} \, ln\left[\frac{P_{11}(\tau_m)}{P_{10}(\tau_m)}\right] \quad , \quad m \geq 2 \tag{8.6}$$

---

[4] It can be shown that the variance of $ln(r_{t+\Delta t})$ conditional to r_t is given by $\hat{V}(\, ln(\, r_{t+\Delta t} \,) \mid r_t \,) = \nu_t^2 \, \Delta t$, where ν_t is the short-rate volatility as seen at time t. Under a binomial representation, the same variance is calculated to be $¼ \, ln^2(\, r_{i+1\,j} / r_{i+1\,j+1} \,)$ conditional to the starting rate of r_{ij}. Thus, the branching rule for r_{ij} should satisfy the condition that

$$(\, r_{i+1\,j+1} / r_{i+1\,j} \,) = (\, d_{ij} / u_{ij} \,) = e^{-2\nu_{ij}\sqrt{\Delta t}}$$

For non-stochastic short-rate volatility that does not depend on r_{ij}, the factor $\beta_i = e^{-2\nu_{ij}\sqrt{\Delta t}}$ will depend only on time.

[5] Refer to the derivation in footnote 4 for bond price volatility in (8.6) and note that it is positively defined.

It is clear that the current short rate $r_{00} = \alpha_0$ at time step $i = 0$ can be fixed by calibrating with $P_0(\tau_1)$ as in (8.5). This gives

$$\alpha_0 = -\frac{1}{\Delta t} lnP_0(\tau_1) \tag{8.7}$$

Knowing r_{00}, the two short rates $r_{10} = \alpha_1$ and $r_{11} = \alpha_1\beta_1$ at time step $i = 1$ can be fixed by calibrating with $P_0(\tau_2)$ and $\sigma_0(\tau_2)$, as in (8.5) and (8.6) respectively. This can be achieved by implementing the Newton-Raphson procedure for $\{ \alpha_1, \beta_1 \}$ taking $\{ \alpha_0, \beta_0 = 0.5 \}$ as their trial values. In (8.2), short rates at time step i are parameterized by only two factors as $r_{ij} = \alpha_i (\beta_i)^j$. Similarly, they can be fixed by calibrating with $P_0(\tau_{i+1})$ and $\sigma_0(\tau_{i+1})$ knowing all previous short rates on the tree. In this way, the tree can be constructed through forward induction in time that subsequently matches current bond prices and volatilities with longer maturity terms. A short-rate tree with horizon $T_{tree} = N_{tree}\Delta t$ would require as input market term structures with maturities $\{ \tau_1, \tau_2, \dots, \tau_{N_{tree}+1} \}$. The factors $\{ \alpha_i, \beta_i \}$ at each stage can be calibrated utilizing the Newton-Raphson procedure with trial values $\{ \alpha_{i-1}, \beta_{i-1} \}$ taken from preceding time step.

A BDT tree can easily be applied to price interest rate derivatives based on the risk-neutral pricing in (8.3). For example, consider a European call option with strike price K and maturity T written on a coupon-bearing bond that matures at later time τ. The bond has a face value of L_{par} and pays regular coupons of value C under the time schedule $\{ s_1, s_2, \dots, s_{n_c} \}$. In this case, we need to construct a BDT tree with time horizon that covers the entire life of the underlying bond. To maximize accuracy in the pricing, the size of the time interval Δt must be chosen very close to the shortest maturity term on bond data while it could also reach τ with discrete time steps. Market term structures with maturities that coincide with such time increments can be constructed through cubic spline interpolation. Strictly speaking, the BDT tree should be constructed up to one time step prior to the maturity of the underlying bond. The total time horizon is thus given by $T_{tree} = \tau - \Delta t$ with $N_{tree} = (\tau/\Delta t) - 1$ steps. For simplicity, suppose the option matures at time step H on the tree for which $T = H\Delta t$. The maturity payoff of the option $\psi(r_{Hj})$ is evaluated according to the realized forward price of the underlying bond on the tree node (H, j) as

$$f_{Hj} = max\{ B_{Hj}(\tau) - K , 0 \} \quad , \quad j = 0, 1, \dots, H \tag{8.8}$$

The forward bond prices in (8.8) can be determined by iterating (8.3) for the underlying bond that utilizes the BDT tree. The iteration starts off from the bond's maturity at time step N_{tree} with face value L_{par} and works backward to the option's maturity at time step H. The coupon payments can be taken as update to the risk-neutral pricing in (8.3) with respect to intermediate boundary condition as

$$B_{ij}(\tau) + \rho_i(s_1, s_2, \dots, s_{n_c})C \tag{8.9}$$

The term $\rho_i(s_1, s_2, \dots, s_{n_c})$ in (8.9) counts the total number of coupons being paid during the time interval $(i - \frac{1}{2})\Delta t < t \le (i + \frac{1}{2})\Delta t$. Thus, the option payoffs on different tree nodes can readily be evaluated, and the current price of the option f_{00} can be determined by iterating (8.3) backward again for the option from time step H to 0.

8.2. EXCEL Plus VBA Implementation

We first develop a routine called GenBDTTree( ) that generates the BDT short-rate tree given current term structures of zero-coupon bond prices and their volatilities. The pseudo code of GenBDTTree( ) is given by Code 8.1. It requires the input of tree configuration (T_{tree} , N_{tree}) and the market term structures with maturities $\{ \tau_1, \tau_2, \dots, \tau_{N_{tree}+1} \}$. The routine returns the array of short rates r_{ij} at every node of the BDT tree with time label i runs from 0 to N_{tree} and node label j runs from

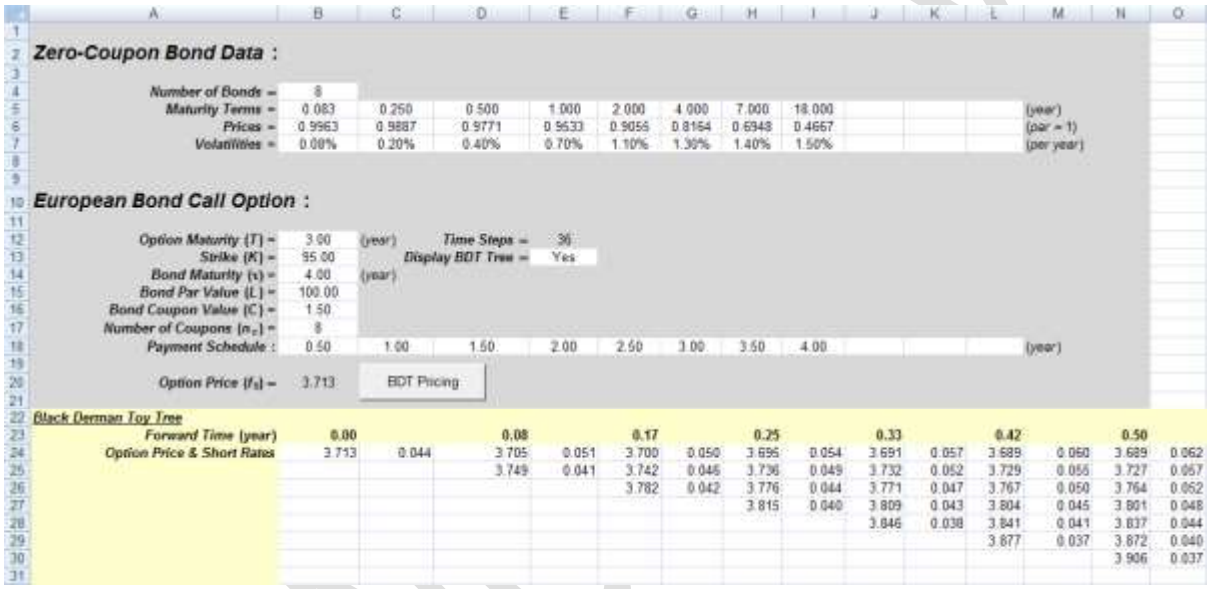
0 to i. The time interval of the tree is defined to be $\Delta t = T_{tree} / N_{tree}$. Together with the input market term structures, they are kept as common data at module scope that can be accessed by other routines within the module. The routine will generate the factors α and β for each column of short rates through forward-time induction regulated by the time pointer *iptr*. The addressed α_{iptr} and β_{iptr} are determined based on their results at preceding time steps from 0 to *iptr* − 1. It is also convenient to declare both *iptr*, α, and β as module-level variables rather than passing their updated values through subroutine arguments.

The starting value α_0 is defined in (8.7) and we arbitrarily choose β_0 to be the mid-value 0.5. To generate the entire BDT tree, the time pointer *iptr* would be required to run from 1 to N_{tree}. At each *iptr*, the tree has presumably been constructed up to time step *iptr* − 1. The factors α_{iptr} and β_{iptr} are determined by calling a two-dimensional Newton-Raphson procedure as discussed in previous chapter. It will solve for $x(1)$ and $x(2)$, setting as α_{iptr} and β_{iptr} respectively, such that the discrepancies in (8.5) and (8.6) for maturity term τ_{iptr+1} are both acceptable under the specified precision *prec*. The trial values of $x(1)$ and $x(2)$ in the numerical search are taken to be the results at preceding time step. The discrepancies, namely $g(1)$ and $g(2)$, are calculated through an external routine Functionarray( ) that iterates all forward bond prices using (8.4) with maturity values of \$1 at *iptr* + 1 and utilizing the trial values at *iptr* together with the results in previous time steps. Here, the values of $P_0(\tau_{iptr+1})$ and $\sigma_0(\tau_{iptr+1})$ on the market term structures as well as the resulting α and β in previous time steps can readily be assessed by virtue of their declaration at module level. Finally, short rates for the entire BDT tree can easily be generated according to (8.2) once all α and β have been determined.

The VBA code of GenBDTTree( ) and Functionarray( ) are given by Code 8.2. Altogether, they are kept under the module called BDTtree[6] and can readily be used to price, for example, the European call option written on a coupon-bearing bond as defined in (8.8). In this respect, we develop a routine called GenBDTBondOptionTree( ) that performs the iteration and generates the option prices along the BDT tree. The pseudo code of GenBDTBondOptionTree( ) is given by Code 8.3. It requires the input of option parameters (T , K , τ , L_{par} , C , n_c , { $s_1, s_2, \ldots, s_{n_c}$ }) and returns the array of option prices f_{ij} as well as the short rates r_{ij} at every tree node prior to the option maturity with time label i runs from 0 to H and node label j runs from 0 to i. The routine first constructs the market term structures with horizon τ that matches the maturity of the underlying bond. These are done through an external procedure called GenTermStructures( ) capable of generating the term structures with maturities { $\tau_1, \tau_2, \ldots, \tau_{N_{term}} = \tau$ } where $\tau_m = m\Delta t$. The size of the time interval Δt has been chosen according to the available bond data as discussed previously. Recall that the BDT tree should be constructed up to one time step prior to the maturity of the bond. We should therefore define the tree configuration to be $N_{tree} = N_{term} - 1$ and $T_{tree} = \tau - \Delta t$. The corresponding BDT tree of short rates can be constructed very easily by calling the GenBDTTree( ) routine. It is then straight forward to use the short rates on the tree and to generate the forward prices of the underlying bond at option maturity with time label H. For convenience, we should generate the bond prices for the entire BDT tree such that the coding can be easily modified to value exotic option with intermediate boundary condition that depends on forward bond price. The forward bond prices $B_{Hj}(\tau)$ can be used to evaluate the option payoffs f_{Hj} at maturity and the option prices f_{ij} at every other tree node can be generated by iterating (8.3) backward in time. The external function CouponCount( ) counts the number of coupons being paid at each time step in the iteration and updates the risk-neutral pricing as discussed in (8.9). In CouponCount( ), we are running through the entire payment schedule { $s_1, s_2, \ldots, s_{n_c}$ } in backward order and identify those payments that appear within the specified time range from t_{low} to t_{up} . It is efficient to exit the procedure by setting *exitFlag* = TRUE whenever we see an unmatched payment immediately after a matched case (when *CouponCount* > 0). The VBA code of GenBDTBondOptionTree( ) is given by Code 8.4.

[6] Refer to BDTtree_ebc.xls

The VBA code of the GenTermStructures( ) routine is given by Code 8.5. It reads in from EXCEL zero-coupon bond prices and volatilities of available maturities and generates the required term structures through cubic spline interpolation. It should be noted that the specific horizon should not exceed the longest maturity term on market bond data. The total number of time steps N_{term} is chosen to be the nearest multiple of the shortest market term in covering the specified horizon. The size of the time interval defined as $\Delta t = (horizon / N_{term})$ will be close to the shortest market term. In fact, it will be greater than the shortest market term as N_{term} is a truncated integer. The cubic spline coefficients can be determined by calling the CubicSpline( ) routine with market bond data. The output term structures with maturities $\{ \Delta t, 2\Delta t, \ldots , N_{term}\Delta t \}$ can then be generated by evaluating the interpolated values using the coefficients based on a similar procedure as discussed in Code 6.5. The routine can also generate the term structures with additional maturities $\{ \Delta t, 2\Delta t, \ldots , (N_{term} + N_a)\Delta t \}$ as defined by the integer N_a in the input. However, the total horizon should remain below the longest maturity term on market bond data.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2	**Zero-Coupon Bond Data :**														
3															
4	Number of Bonds =	8													
5	Maturity Terms =	0.083	0.250	0.500	1.000	2.000	4.000	7.000	18.000				(year)		
6	Prices =	0.9963	0.9887	0.9771	0.9533	0.9055	0.8164	0.6948	0.4667				(par = 1)		
7	Volatilities =	0.08%	0.20%	0.40%	0.70%	1.10%	1.30%	1.40%	1.50%				(per year)		
8															
9															
10	**European Bond Call Option :**														
11															
12	Option Maturity (T) =	3.00	(year)	Time Steps =	36										
13	Strike (K) =	95.00		Display BDT Tree =	Yes										
14	Bond Maturity (s) =	4.00	(year)												
15	Bond Par Value (L) =	100.00													
16	Bond Coupon Value (C) =	1.50													
17	Number of Coupons (n_c) =	8													
18	Payment Schedule :	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00				(year)		
19															
20	Option Price (f_{00}) =	3.713	BDT Pricing												
21															
22	Black Derman Toy Tree														
23	Forward Time (year)	0.00		0.08		0.17		0.25		0.33		0.42		0.50	
24	Option Price & Short Rates	3.713	0.044	3.705	0.051	3.700	0.050	3.695	0.054	3.691	0.057	3.689	0.060	3.689	0.062
25				3.749	0.041	3.742	0.046	3.736	0.049	3.732	0.052	3.729	0.055	3.727	0.057
26						3.782	0.042	3.776	0.044	3.771	0.047	3.767	0.050	3.764	0.052
27								3.815	0.040	3.809	0.043	3.804	0.045	3.801	0.048
28										3.846	0.038	3.841	0.041	3.837	0.044
29												3.877	0.037	3.872	0.040
30														3.906	0.037
31															

Figure 8.2: Spreadsheet design of Black-Derman-Toy option pricing.

Figure 8.2 depicts the spreadsheet design for this VBA implementation. The market bond data are inputted into the GenTermStructures( ) routine through the named cell B4(nbond) and the entries in rows 5, 6, and 7. Presumably, these could be obtained utilizing the bootstrapped zero rates from the CalZeroRates( ) routine as discussed in previous chapter. The button labeled as "BDT Pricing" will trigger the main VBA routine called BDTPricing( ) with VBA code given by Code 8.6. The option parameters are inputted into this routine through the named cells B12(optionMaturity), B13(strike), B14(bondMaturity), B15(par), B16(coupon), B17(nCoupon), and the entries in row 18. It will call GenBDTBondOptionTree( ) for the arrays of option prices and short rates. The current option price f_{00} will be outputted to cell B20. The entire option trees will be displayed optionally in spreadsheet with respect to the choice of "Yes" in cell E13. As reference, the corresponding short rates will also be displayed adjacent to the option prices. Similar to the procedure as discussed in Code 7.3, this can be done by running over both the time and node labels, and allocating cells through offset from the reference cell B23. Recall that there are all together 255 columns in spreadsheet. The option maturity in B12 will determine the number of time steps on the tree as displayed in E12 = **INT**(B12 / B5). It should be taken below 127 when we choose "Yes" in E13. It is then necessary to impose a validation check for the cell B12 as

```
= IF( AND( E13 = "Yes" , E12 > = 127 ) , FALSE , TRUE)
```

under **Data**, **Validation**, and **Setting**s.

---

define the following module-level variables
Δt , P_0(1 : *Ntreemax* + 1) , σ_0(1 : *Ntreemax* + 1) , *iptr* , α(0 : *Ntreemax*) , β(0 : *Ntreemax*)

GenBDTTree(T_{tree} , N_{tree} , *BondPrice*(1 : N_{tree} + 1) , *BondVol*(1 : N_{tree} + 1) , *r*(0 : N_{tree} , 0 : N_{tree}))

define the size of the time interval
$\Delta t = T_{tree} / N_{tree}$

# assign the market term structures with maturities τ_1, τ_2, … , and $\tau_{N_{tree}+1}$ to common arrays
For($k = 1$ to $N_{tree} + 1$) { $P_0(k) = BondPrice(k)$, $\sigma_0(k) = BondVol(k)$ }

define α_0 according to (8.7) and choose β_0 to be ½
$\alpha(0) = (-1/\Delta t)\ logP_0(1)$, $\beta(0) = 0.5$

set *x*(1) and *x*(2) as α and β, respectively. Use the Newton-Raphson procedure to estimate next *x*(1)
and *x*(2) taking the last results as trial values.
$x(1) = \alpha(0)$, $x(2) = \beta(0)$
For(*iptr* = 1 to N_{tree}) { Call NewtonRaphson(2, *prec* , *x*(1 : 2) , *precflag* , *maxdev*)
α(*iptr*) = *x*(1) , β(*iptr*) = *x*(2) }

# generate the short-rate tree from the resulting α and β according to (8.2)
For($k = 0$ to N_{tree}) { For($j = 0$ to k){ $r(k, j) = \alpha(k)\ \beta(k)^j$ } }

Functionarray(*n* , *x*(1 : *n*) , *g*(1 : *n*))

# define the face values of the zero-coupon bond with maturity at τ_{i+1}
For($j = 0$ to *iptr* + 1) { $P_{forward}$(*iptr* + 1 , *j*) = 1 }

iterate (8.4) backward in time and generate all forward bond prices on the tree using the trial values
of $\alpha_i = x(1)$ and $\beta_i = x(2)$ together with known α and β in previous time steps

For($j = 0$ to *iptr*) {
$P_{forward}$(*iptr* , *j*) = $e^{-x(1)\,x(2)^j\,\Delta t}$ [½ $P_{forward}$(*iptr* + 1 , *j*) + ½ $P_{forward}$(*iptr* + 1 , *j* + 1)] }

For(k = *iptr* − 1 to 0, −1) {
For($j = 0$ to k) { $P_{forward}$(k , j) = $e^{-\alpha(k)\beta(k)^j\,\Delta t}$ [½ $P_{forward}$($k + 1$, j) + ½ $P_{forward}$($k + 1$, $j + 1$)] }
}

calculate the discrepancies in (8.5) and (8.6)
$g(1) = P_0(iptr + 1) - P_{forward}(0, 0)$

$$g(2) = \sigma_0(iptr + 1) - \frac{1}{2\sqrt{\Delta t}}\ ln\left[\frac{P_{forward}(1, 1)}{P_{forward}(1, 0)}\right]$$

Code 8.1 : Pseudo code of the GenBDTTree( ) routine.

```
Option Explicit
Private iptr As Integer
Private dtime As Double
Private alpha(0 To nTreeMax) As Double
Private beta(0 To nTreeMax) As Double
Private P0(1 To nTreeMax + 1) As Double
Private Sigma0(1 To nTreeMax + 1) As Double
```

```
Sub GenBDTTree(Ttree As Double, Ntree As Integer, BondPrice() As Double, BondVol() As Double, rshort() As Double)
  Dim x(1 To 2) As Double, prec As Double, precFlag As Boolean, maxDev As Double
  prec = 0.00000001

  Dim k As Integer, j As Integer

  Dim dtime As Double: dtime = Ttree / Ntree

  For k = 1 To Ntree + 1
    P0(k) = BondPrice(k)
    Sigma0(k) = BondVol(k)
  Next k

  alpha(0) = -(1 / dtime) * Log(P0(1))
  beta(0) = 0.5

  x(1) = alpha(0)
  x(2) = beta(0)

  For iptr = 1 To Ntree
    Call NewtonRaphson(2, prec, x, precFlag, maxDev)
    alpha(iptr) = x(1)
    beta(iptr) = x(2)
  Next iptr

  For k = 0 To Ntree
    For j = 0 To k: rshort(k, j) = alpha(k) * beta(k) ^ j: Next j
  Next k

End Sub
```

```
Sub FunctionArray(n As Integer, x() As Double, ByRef g() As Double)
  Dim Pf(0 To nTreeMax + 1, 0 To nTreeMax + 1) As Double
  Dim k As Integer, j As Integer

  For j = 0 To iptr + 1: Pf(iptr + 1, j) = 1: Next j

  For j = 0 To iptr
    Pf(iptr, j) = Exp(-x(1) * x(2) ^ j * dtime) * (Pf(iptr + 1, j) + Pf(iptr + 1, j + 1)) / 2
  Next j

  For k = iptr - 1 To 0 Step -1
    For j = 0 To k
      Pf(k, j) = Exp(-alpha(k) * beta(k) ^ j * dtime) * (Pf(k + 1, j) + Pf(k + 1, j + 1)) / 2
    Next j
  Next k

  g(1) = P0(iptr + 1) - Pf(0, 0)
  g(2) = Sigma0(iptr + 1) - Log(Pf(1, 1) / Pf(1, 0)) / (2 * Sqr(dtime))

End Sub
```

Code 8.2 : VBA code of the GenBDTTree( ) routine together with the Functionarray( ) routine.

GenBDTBondOptionTree(T , K , τ , L_{par} , C , n_c , $s(1:n_c)$, H , $r(0:H, 0:H)$, $f(0:H, 0:H)$)

generate the market term structures with horizon τ
Call GenTermStructures(τ , 0 , N_{term} , Δt , *BondPrice*($1:N_{term}$) , *BondVol*($1:N_{term}$))

define BDT tree configuration
$N_{tree} = N_{term} - 1$, $T_{tree} = \tau - \Delta t$

generate the BDT tree with N_{tree} steps and horizon T_{tree}
Call GenBDTTree(T_{tree} , N_{tree} , *BondPrice*($1:N_{tree}+1$) , *BondVol*($1:N_{tree}+1$) , $r(0:N_{tree}, 0:N_{tree})$)

define the time label at option maturity
$H = Int(T / \Delta t)$

generate the forward prices of the underlying bond

ρ = *CouponCount*($(N_{tree}+1-½)\Delta t$, $(N_{tree}+1+½)\Delta t$, n_c , $s(1:n_c)$)
For($j = 0$ to $N_{tree}+1$) { $B_{forward}(N_{tree}+1, j) = L_{par} + \rho C$ }

For($i = N_{tree}$ to 0 , −1) {
  ρ = *CouponCount*($(i-½)\Delta t$, $(i+½)\Delta t$, n_c , $s(1:n_c)$)
  For($j = 0$ to i) { $B_{forward}(i, j) = e^{-r(i,j)\Delta t}[½ B_{forward}(i+1, j) + ½ B_{forward}(i+1, j+1)] + \rho C$ }
}

generate the option prices

For($j = 0$ to H) { $f(H, j) = Payoff(K, B_{forward}(H, j))$ }

For($i = H - 1$ to 0, −1) {
  For($j = 0$ to i) { $f(i, j) = e^{-r(i,j)\Delta t}[½ f(i+1, j) + ½ f(i+1, j+1)]$ }
}

CouponCount(t_{low} , t_{up} , n_c , $s(1:n_c)$)

exitFlag = FALSE
CouponCount = 0
For($k = n_c$ to 1, −1) { If ($t_{low} < s(k) \le t_{up}$) Then
  CouponCount = *CouponCount* + 1
Elseif (*CouponCount* > 0) Then
  exitFlag = TRUE
Endif
If(*exitFlag*) Exit k }

Code 8.3 : Pseudo code of the GenBDTBondOptionTree( ) routine.

```
Sub GenBDTBondOptionTree(optionMaturity As Double, strike As Double, bondMaturity As Double, par As Double, coupon As Double,
                         nCoupon As Integer, paymentSchedule() As Double, ByRef Hf As Integer, ByRef rShort() As Double,
                         ByRef fTree() As Double)

  Dim bondPrice(1 To nTreeMax + 1) As Double
  Dim bondVol(1 To nTreeMax + 1) As Double
  Dim i As Integer, j As Integer
  Dim Nterm As Integer, dtime As Double

  Call GenTermStructures(bondMaturity, 0, Nterm, dtime, bondPrice, bondVol)

  Dim Ntree As Integer: Ntree = Nterm - 1
  Dim Ttree As Double: Ttree = bondMaturity - dtime

  Call GenBDTTree(Ttree, Ntree, bondPrice, bondVol, rShort)

  Dim Bf() As Double: ReDim Bf(0 To Ntree + 1, 0 To Ntree + 1)

  Hf = Int(optionMaturity / dtime)

  i = Ntree + 1
  Dim rho As Integer
  rho = CouponCount((i - 0.5) * dtime, (i + 0.5) * dtime, nCoupon, paymentSchedule)
  For j = 0 To i: Bf(i, j) = par + rho * coupon: Next j

  For i = Ntree To 0 Step -1
    rho = CouponCount((i - 0.5) * dtime, (i + 0.5) * dtime, nCoupon, paymentSchedule)
    For j = 0 To i
      Bf(i, j) = Exp(-rShort(i, j) * dtime) * (Bf(i + 1, j) + Bf(i + 1, j + 1)) / 2 + rho * coupon
    Next j
  Next i

  For j = 0 To Hf: fTree(Hf, j) = Payoff(strike, Bf(Hf, j)): Next j

  For i = Hf - 1 To 0 Step -1
    For j = 0 To i
      fTree(i, j) = Exp(-rShort(i, j) * dtime) * (fTree(i + 1, j) + fTree(i + 1, j + 1)) / 2
    Next j
  Next i

End Sub
```

```
Function CouponCount(timeLow As Double, timeUp As Double, nCoupon As Integer, paymentSchedule() As Double) As Integer
  Dim k As Integer
  Dim exitFlag As Boolean

  exitFlag = False
  CouponCount = 0

  For k = nCoupon To 1 Step -1
    If (Round(paymentSchedule(k), epsDP) > timeLow And Round(paymentSchedule(k), epsDP) <= timeUp) Then
      CouponCount = CouponCount + 1
    ElseIf (CouponCount > 0) Then
      exitFlag = True
    End If
    If (exitFlag) Then Exit For
  Next k

End Function
```

Code 8.4 : VBA code of the GenBDTBondOptionTree( ) routine.

```
Sub GenTermStructures(horizon As Double, Na As Integer, ByRef Nterm As Integer, ByRef dtime As Double,
                      ByRef bondPrice() As Double, ByRef bondVol() As Double)

  Dim nbond As Integer: nbond = Range("nbond").Value
  Dim bondMaturity() As Double: ReDim bondMaturity(1 To nbond)
  Dim bondPriceData() As Double: ReDim bondPriceData(1 To nbond)
  Dim bondVolData() As Double: ReDim bondVolData(1 To nbond)
  Dim a() As Double: ReDim a(1 To nbond - 1)
  Dim b() As Double: ReDim b(1 To nbond - 1)
  Dim c() As Double: ReDim c(1 To nbond - 1)
  Dim d() As Double: ReDim d(1 To nbond - 1)
  Dim av() As Double: ReDim av(1 To nbond - 1)
  Dim bv() As Double: ReDim bv(1 To nbond - 1)
  Dim cv() As Double: ReDim cv(1 To nbond - 1)
  Dim dv() As Double: ReDim dv(1 To nbond - 1)
  Dim i As Integer, k As Integer

  For i = 1 To nbond
    bondMaturity(i) = Range("A5").Offset(0, i)
    bondPriceData(i) = Range("A6").Offset(0, i)
    bondVolData(i) = Range("A7").Offset(0, i)
  Next i

  Call CubicSpline(nbond, bondMaturity, bondPriceData, a, b, c, d)
  Call CubicSpline(nbond, bondMaturity, bondVolData, av, bv, cv, dv)

  Nterm = Int(horizon / bondMaturity(1))
  dtime = horizon / Nterm

  Dim term As Double
  Dim ptr As Integer

  ptr = 1
  For i = 1 To Nterm + Na
    term = i * dtime
    If (Abs(term - bondMaturity(1)) <= eps) Then
      bondPrice(i) = bondPriceData(1)
      bondVol(i) = bondVolData(1)
      GoTo Nexti
    End If
    For k = ptr To nbond - 1
      If (term > bondMaturity(k) And term <= bondMaturity(k + 1)) Then
        bondPrice(i) = a(k) + b(k) * term + c(k) * term ^ 2 + d(k) * term ^ 3
        bondVol(i) = av(k) + bv(k) * term + cv(k) * term ^ 2 + dv(k) * term ^ 3
        ptr = k
        GoTo Nexti
      End If
    Next k
Nexti:  Next i

End Sub
```

Code 8.5 : VBA code of the GenTermStructures( ) routine.

```
Sub BDTPricing()
  Dim i As Integer, j As Integer
  Dim rShort(0 To nTreeMax, 0 To nTreeMax) As Double
  Dim fTree(0 To nTreeMax, 0 To nTreeMax) As Double
  Dim Hf As Integer

  Dim optionMaturity As Double: optionMaturity = Range("optionMaturity").Value
  Dim strike As Double: strike = Range("strike").Value
  Dim bondMaturity As Double: bondMaturity = Range("bondMaturity").Value
  Dim par As Double: par = Range("par").Value
  Dim coupon As Double: coupon = Range("coupon").Value
  Dim nCoupon As Integer: nCoupon = Range("nCoupon").Value
  Dim paymentSchedule() As Double: ReDim paymentSchedule(0 To nCoupon)

  For i = 1 To nCoupon: paymentSchedule(i) = Range("A18").Offset(0, i): Next i

  Call GenBDTBondOptionTree(optionMaturity, strike, bondMaturity, par, coupon, nCoupon, paymentSchedule, Hf, rShort, fTree)
  Range("B20").Value = fTree(0, 0)

  Range("B23:IV150").ClearContents

  If (Range("E13").Text = "Yes") Then
    For i = 0 To Hf
      Range("B23").Offset(0, 2 * i) = i * (optionMaturity / Hf)
      For j = 0 To i
        Range("B23").Offset(j + 1, 2 * i) = fTree(i, j)
        Range("B23").Offset(j + 1, 2 * i + 1) = rShort(i, j)
      Next j
    Next i
  End If

End Sub
```

Code 8.6 : VBA code of the BDTPricing( ) routine.