Backward Induction on the RT Tree

- After the RT tree is constructed, it can be used to price options by backward induction.
- Recall that each node keeps two variances $h^2_{\max}$ and $h^2_{\min}$.
- We now increase that number to K equally spaced variances between $h^2_{\max}$ and $h^2_{\min}$ at each node.
- Besides the minimum and maximum variances, the other $K - 2$ variances in between are linearly interpolated.[a]

[a] In practice, log-linear interpolation works better (Lyuu and Wu (R90723065) (2005)). Log-cubic interpolation works even better (Liu (R92922123) (2005)).

Backward Induction on the RT Tree (continued)

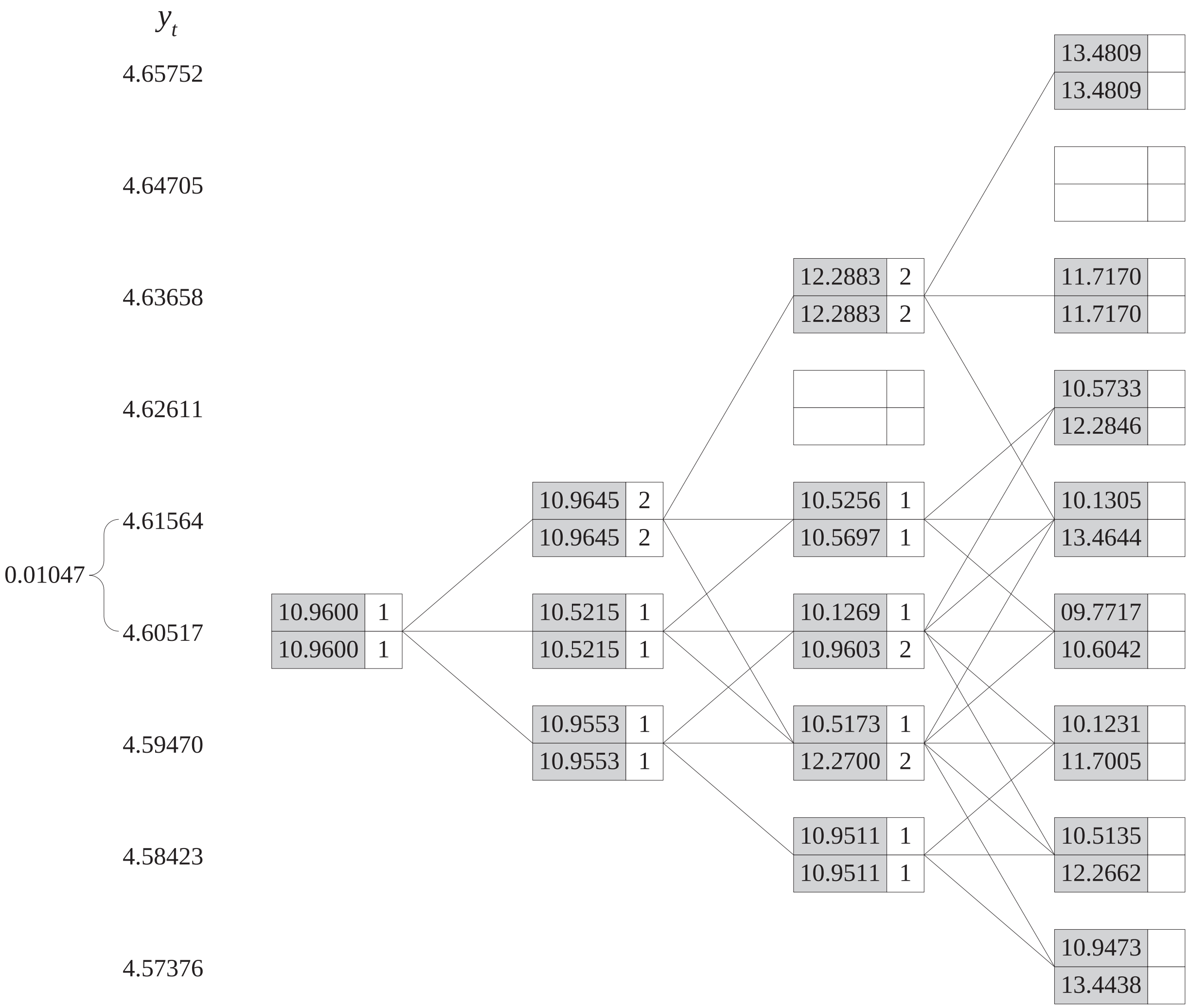
- For example, if $K = 3$, then a variance of 10.5436×10^{-6} will be added between the maximum and minimum variances at node $(2, 0)$ on p. 768.[a]
- In general, the kth variance at node (i, j) is

$$h^2_{\min}(i, j) + k \, \frac{h^2_{\max}(i, j) - h^2_{\min}(i, j)}{K - 1},$$

$k = 0, 1, \ldots, K - 1$.

- Each interpolated variance's jump parameter and branching probabilities can be computed as before.

[a] Repeated on p. 788.

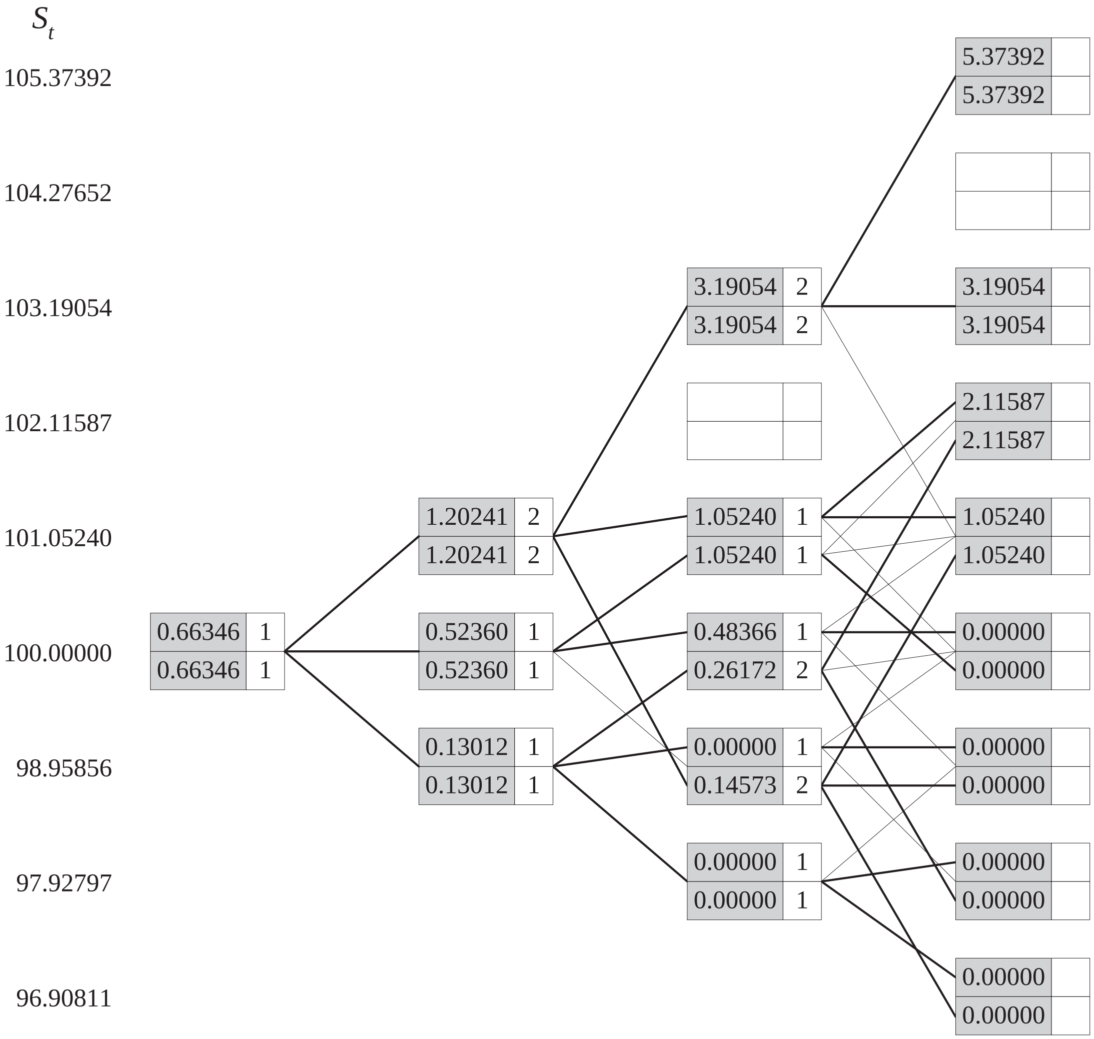

y_t
4.65752
4.64705
4.63658
4.62611
4.61564
0.01047
4.60517
4.59470
4.58423
4.57376
10.9600 1
10.9600 1
10.9645 2
10.9645 2
10.5215 1
10.5215 1
10.9553 1
10.9553 1
12.2883 2
12.2883 2
10.5256 1
10.5697 1
10.1269 1
10.9603 2
10.5173 1
12.2700 2
10.9511 1
10.9511 1
13.4809
13.4809
11.7170
11.7170
10.5733
12.2846
10.1305
13.4644
09.7717
10.6042
10.1231
11.7005
10.5135
12.2662
10.9473
13.4438

Backward Induction on the RT Tree (concluded)

- Suppose a variance falls between two of the K variances during backward induction.
- Linear interpolation of the option prices corresponding to the two bracketing variances will be used as the approximate option price.
- The above ideas are reminiscent of the ones on p. 349, where we dealt with arithmetic average-rate options.

Numerical Examples

- We next use the numerical example on p. 788 to price a European call option with a strike price of 100 and expiring at date 3.
- Recall that the riskless interest rate is zero.
- Assume $K = 2$; hence there are no interpolated variances.
- The pricing tree is shown on p. 791 with a call price of 0.66346.
  - The branching probabilities needed in backward induction can be found on p. 792.


S_t
105.37392
104.27652
103.19054
102.11587
101.05240
100.00000
98.95856
97.92797
96.90811
0.66346 1
0.66346 1
1.20241 2
1.20241 2
0.52360 1
0.52360 1
0.13012 1
0.13012 1
3.19054 2
3.19054 2
1.05240 1
1.05240 1
0.48366 1
0.26172 2
0.00000 1
0.14573 2
0.00000 1
0.00000 1
5.37392
5.37392
3.19054
3.19054
2.11587
2.11587
1.05240
1.05240
0.00000
0.00000
0.00000
0.00000
0.00000
0.00000
0.00000
0.00000

rb[i][0]
rb[i][1]

rb[0][ ]	rb[1][ ]	rb[2][ ]	rb[3][ ]
0 0	−1 1	−2 3	−3 5

h^2[i][j][0]
h^2[i][j][1]

h^2[0][ ][ ]	h^2[1][ ][ ]	h^2[2][ ][ ]	h^2[3][ ][ ]	j
			13.4809 13.4809	5
				4
		12.2883 12.2883	11.7170 11.7170	3
			10.5733 12.2846	2
	10.9645 10.9645	10.5256 10.5697	10.1305 13.4644	1
10.9600 10.9600	10.5215 10.5215	10.1269 10.9603	09.7717 10.6042	0
	10.9553 10.9553	10.5173 12.2700	10.1231 11.7005	−1
		10.9511 10.9511	10.5135 12.2662	−2
			10.9473 13.4438	−3

η[i][j][0]
η[i][j][1]

η[0][ ][ ]	η[1][ ][ ]	η[2][ ][ ]	j
		2 2	3
			2
	2 2	1 1	1
1 1	1 1	1 2	0
	1 1	1 2	−1
		1 1	−2

p[i][j][0][1]	p[i][j][1][1]
p[i][j][0][0]	p[i][j][1][0]
p[i][j][0][−1]	p[i][j][1][−1]

p[0][ ][ ][ ]	p[1][ ][ ][ ]	p[2][ ][ ][ ]	j
		0.1387 0.1387 0.7197 0.7197 0.1416 0.1416	3
			2
	0.1237 0.1237 0.7499 0.7499 0.1264 0.1264	0.4777 0.4797 0.0396 0.0356 0.4827 0.4847	1
0.4974 0.4974 0.0000 0.0000 0.5026 0.5026	0.4775 0.4775 0.0400 0.0400 0.4825 0.4825	0.4596 0.1237 0.0760 0.7500 0.4644 0.1263	0
	0.4972 0.4972 0.0004 0.0004 0.5024 0.5024	0.4773 0.1385 0.0404 0.7201 0.4823 0.1414	−1
		0.4970 0.4970 0.0008 0.0008 0.5022 0.5022	−2

Numerical Examples (continued)

- Let us derive some of the numbers on p. 791.
- A gray line means the updated variance falls strictly between $h^2_{\max}$ and $h^2_{\min}$.
- The option price for a terminal node at date 3 equals $\max(S_3 - 100, 0)$, independent of the variance level.
- Now move on to nodes at date 2.
- The option price at node $(2, 3)$ depends on those at nodes $(3, 5)$, $(3, 3)$, and $(3, 1)$.
- It therefore equals

$$0.1387 \times 5.37392 + 0.7197 \times 3.19054 + 0.1416 \times 1.05240 = 3.19054.$$

Numerical Examples (continued)

- Option prices for other nodes at date 2 can be computed similarly.
- For node $(1, 1)$, the option price for both variances is

$$0.1237 \times 3.19054 + 0.7499 \times 1.05240 + 0.1264 \times 0.14573 = 1.20241.$$

- Node $(1, 0)$ is most interesting.
- We knew that a down move from it gives a variance of 0.000105609.
- This number falls between the minimum variance 0.000105173 and the maximum variance 0.0001227 at node $(2, -1)$ on p. 788.

Numerical Examples (continued)

- The option price corresponding to the minimum variance is 0.
- The option price corresponding to the maximum variance is 0.14573.
- The equation

$$x \times 0.000105173 + (1 - x) \times 0.0001227 = 0.000105609$$

is satisfied by $x = 0.9751$.

- So the option for the down state is approximated by

$$x \times 0 + (1 - x) \times 0.14573 = 0.00362.$$

Numerical Examples (continued)

- The up move leads to the state with option price 1.05240.
- The middle move leads to the state with option price 0.48366.
- The option price at node $(1, 0)$ is finally calculated as

$$0.4775 \times 1.05240 + 0.0400 \times 0.48366 + 0.4825 \times 0.00362 = 0.52360.$$

Numerical Examples (continued)

- It is possible for some of the three variances following an interpolated variance to exceed the maximum variance or be exceeded by the minimum variance.
- When this happens, the option price corresponding to the maximum or minimum variance will be used during backward induction.[a]

[a] Cakici and Topyan (2000).

Numerical Examples (concluded)

- An interpolated variance may choose a branch that goes into a node that is *not* reached in forward induction.[a]
- In this case, the algorithm fails.
- It may also be hard to calculate the implied β_1 and β_2 from option prices.[b]

[a] Lyuu and Wu (R90723065) (2005).

[b] Chang (R93922034) (2006).

Complexities of GARCH Models[a]

- The Ritchken-Trevor algorithm explodes exponentially if n is big enough (p. 765).
- The mean-tracking algorithm of Lyuu and Wu (2005) will make sure explosion does not happen if n is not too large.[b]
- The next page summarizes the situations for many GARCH option pricing models.
  - Our earlier treatment is for NGARCH.

[a] Lyuu and Wu (`R90723065`) (2003, 2005).

[b] Similar to the idea behind the binomial-trinomial tree on pp. 596ff.

Complexities of GARCH Models (concluded)[a]

Model	Explosion	Non-explosion
NGARCH	$\beta_1 + \beta_2 n > 1$	$\beta_1 + \beta_2(\sqrt{n} + \lambda + c)^2 \leq 1$
LGARCH	$\beta_1 + \beta_2 n > 1$	$\beta_1 + \beta_2(\sqrt{n} + \lambda)^2 \leq 1$
AGARCH	$\beta_1 + \beta_2 n > 1$	$\beta_1 + \beta_2(\sqrt{n} + \lambda)^2 \leq 1$
GJR-GARCH	$\beta_1 + \beta_2 n > 1$	$\beta_1 + (\beta_2 + \beta_3)(\sqrt{n} + \lambda)^2 \leq 1$
TS-GARCH	$\beta_1 + \beta_2 \sqrt{n} > 1$	$\beta_1 + \beta_2(\lambda + \sqrt{n}) \leq 1$
TGARCH	$\beta_1 + \beta_2 \sqrt{n} > 1$	$\beta_1 + (\beta_2 + \beta_3)(\lambda + \sqrt{n}) \leq 1$
Heston-Nandi	$\beta_1 + \beta_2(c - \frac{1}{2})^2 > 1$ & $c \leq \frac{1}{2}$	$\beta_1 + \beta_2 c^2 \leq 1$
VGARCH	$\beta_1 + (\beta_2/4) > 1$	$\beta_1 \leq 1$

[a] Chen (R95723051) (2008).

Introduction to Term Structure Modeling

The fox often ran to the hole
by which they had come in,
to find out if his body was still thin enough
to slip through it.
— *Grimm's Fairy Tales*

And the worst thing you can have
is models and spreadsheets.
— Warren Buffet, May 3, 2008

Outline

- Use the binomial interest rate tree to model stochastic term structure.
  - Illustrates the basic ideas underlying future models.
  - Applications are generic in that pricing and hedging methodologies can be easily adapted to other models.
- Although the idea is similar to the earlier one used in option pricing, the current task is more complicated.
  - The evolution of an entire term structure, not just a single stock price, is to be modeled.
  - Interest rates of various maturities cannot evolve arbitrarily, or arbitrage profits may occur.

Issues

- A stochastic interest rate model performs two tasks.
  - Provides a stochastic process that defines future term structures without arbitrage profits.
  - "Consistent" with the observed term structures.
- The unbiased expectations theory, the liquidity preference theory, and the market segmentation theory can all be made consistent with the model.

History

- Methodology founded by Merton (1970).
- Modern interest rate modeling is often traced to 1977 when Vasicek and Cox, Ingersoll, and Ross developed simultaneously their influential models.
- Early models have fitting problems because they may not price today's benchmark bonds correctly.
- An alternative approach pioneered by Ho and Lee (1986) makes fitting the market yield curve mandatory.
- Models based on such a paradigm are called (somewhat misleadingly) arbitrage-free or no-arbitrage models.

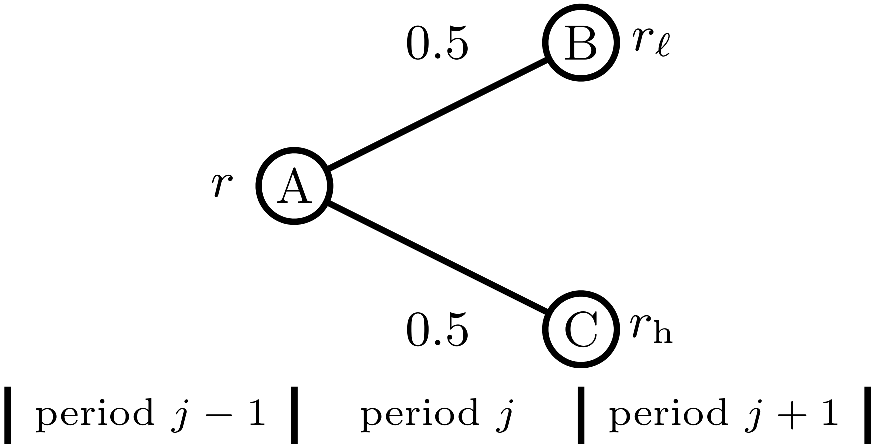
Binomial Interest Rate Tree

- Goal is to construct a no-arbitrage interest rate tree consistent with the yields and/or yield volatilities of zero-coupon bonds of all maturities.
  - This procedure is called calibration.[a]
- Pick a binomial tree model in which the logarithm of the future short rate obeys the binomial distribution.
  - Exactly like the CRR tree.
- The limiting distribution of the short rate at any future time is hence lognormal.

[a] Derman (2004), "complexity without calibration is pointless."

Binomial Interest Rate Tree (continued)

- A binomial tree of future short rates is constructed.
- Every short rate is followed by two short rates in the following period (p. 809).
- In the figure on p. 809 node A coincides with the start of period j during which the short rate r is in effect.


0.5
B
r_ℓ
r
A
0.5
C
$r_{\rm h}$
period $j-1$
period j
period $j+1$

Binomial Interest Rate Tree (continued)

- 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 at node B.
  - r_{h}: the “high” short-rate outcome at node C.
- Each branch has a fifty percent chance of occurring in a risk-neutral economy.

Binomial Interest Rate Tree (continued)

- We shall require that the paths combine as the binomial process unfolds.
- The short rate r can go to r_{h} and r_{ℓ} with equal risk-neutral probability $1/2$ in a period of length Δt.
- Hence the volatility of $\ln r$ after Δt time is

$$\sigma = \frac{1}{2} \frac{1}{\sqrt{\Delta t}} \ln \left(\frac{r_{\mathrm{h}}}{r_{\ell}} \right)$$

  (see Exercise 23.2.3 in text).
- Above, σ is annualized, whereas r_{ℓ} and r_{h} are period based.

Binomial Interest Rate Tree (continued)

- Note that
$$\frac{r_{\rm h}}{r_\ell} = e^{2\sigma\sqrt{\Delta t}}.$$
- Thus greater volatility, hence uncertainty, leads to larger $r_{\rm h}/r_\ell$ and wider ranges of possible short rates.
- The ratio $r_{\rm h}/r_\ell$ may depend on time if the volatility is a function of time.
- Note that $r_{\rm h}/r_\ell$ has nothing to do with the current short rate r if σ is independent of r.

Binomial Interest Rate Tree (continued)

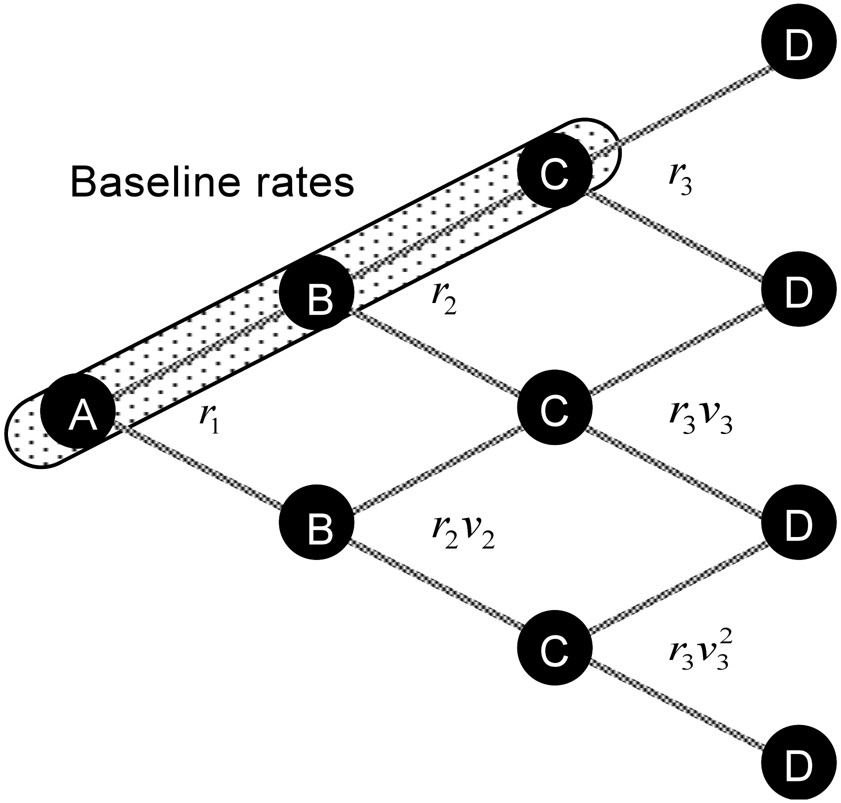
- In general there are j possible rates in period j,

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

  where

  $$v_j \equiv e^{2\sigma_j \sqrt{\Delta t}} \tag{89}$$

  is the multiplicative ratio for the rates in period j (see figure on next page).
- We shall call r_j the baseline rates.
- The subscript j in σ_j is meant to emphasize that the short rate volatility may be time dependent.


D
Baseline rates
C
r_3
D
B
r_2
A
r_1
C
$r_3 v_3$
B
$r_2 v_2$
D
C
$r_3 v_3^2$
D

Binomial Interest Rate Tree (concluded)

- In the limit, the short rate follows the following process,

$$r(t) = \mu(t)\, e^{\sigma(t)\, W(t)}, \tag{90}$$

  in which the (percent) short rate volatility $\sigma(t)$ is a deterministic function of time.

- The expected value of $r(t)$ equals $\mu(t)\, e^{\sigma(t)^2 (t/2)}$.
- Hence a declining short rate volatility is usually imposed to preclude the short rate from assuming implausibly high values.
- Incidentally, this is how the binomial interest rate tree achieves mean reversion.

Memory Issues

- Path independency: The term structure at any node is independent of the path taken to reach it.
- So only the baseline rates r_i and the multiplicative ratios v_i need to be stored in computer memory.
- This takes up only $O(n)$ space.[a]
- Storing the whole tree would take up $O(n^2)$ space.
  - Daily interest rate movements for 30 years require roughly $(30 \times 365)^2/2 \approx 6 \times 10^7$ double-precision floating-point numbers (half a gigabyte!).

[a] Throughout this chapter, n denotes the depth of the tree.

Set Things in Motion

- The abstract process is now in place.
- Now need the annualized rates of return associated with the various riskless bonds that make up the benchmark yield curve and their volatilities.
- In the U.S., for example, the on-the-run yield curve obtained by the most recently issued Treasury securities may be used as the benchmark curve.

Set Things in Motion (concluded)

- The term structure of (yield) volatilities[a] can be estimated from:
  - Historical data (historical volatility).
  - Or interest rate option prices such as cap prices (implied volatility).
- The binomial tree should be consistent with both term structures.
- Here we focus on the term structure of interest rates.

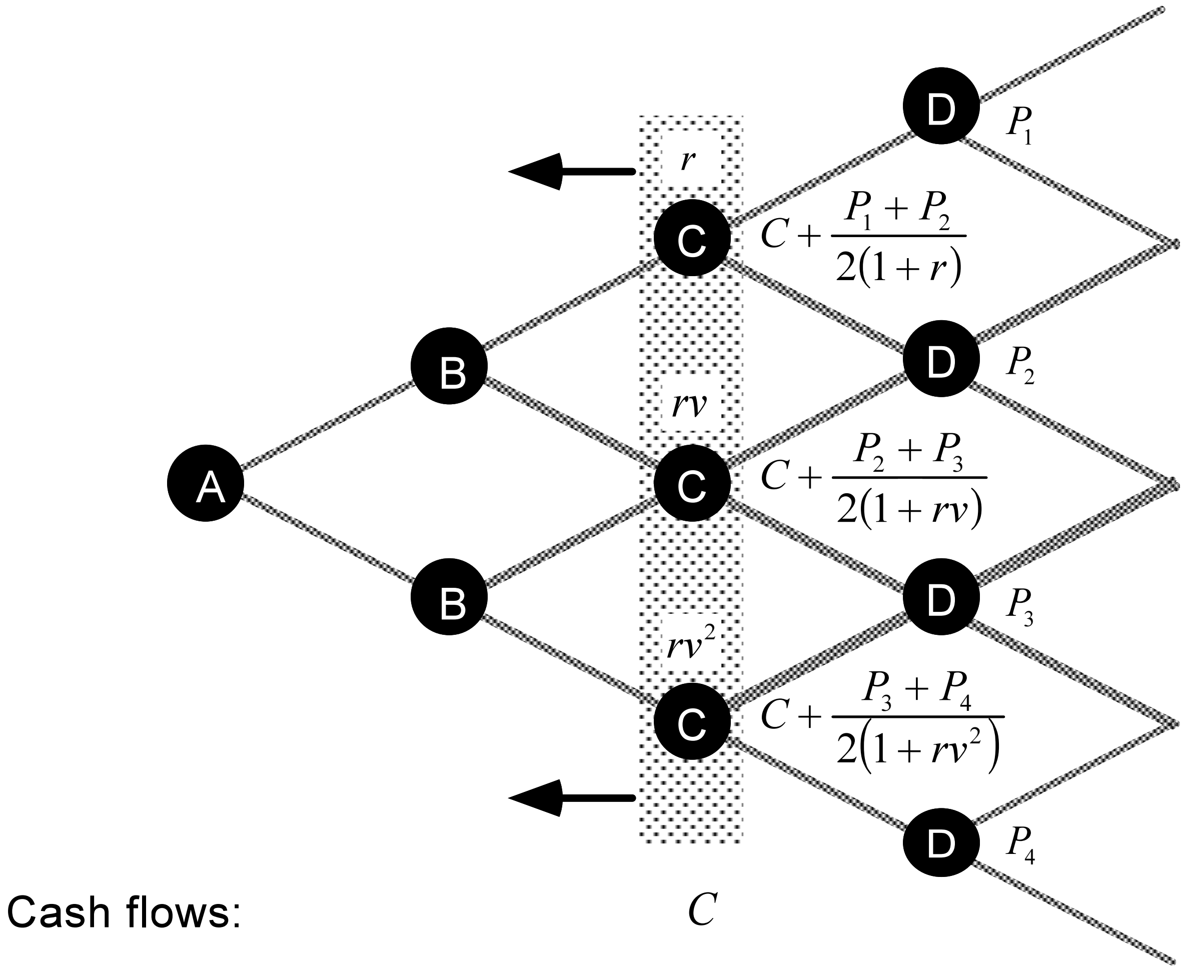
[a] Or simply the volatility (term) structure.

Model Term Structures

- The model price is computed by backward induction.
- Refer back to the figure on p. 809.
- Given that the values at nodes B and C are P_{B} and P_{C}, respectively, the value at node A is then

$$\frac{P_{\mathrm{B}}+P_{\mathrm{C}}}{2(1+r)}+\text{cash flow at node A.}$$

- We compute the values column by column without explicitly expanding the binomial interest rate tree (see figure next page).
- This takes quadratic time and linear space.


A
B
B
r
rv
rv^2
C
C
C
$C + \frac{P_1 + P_2}{2(1+r)}$
$C + \frac{P_2 + P_3}{2(1+rv)}$
$C + \frac{P_3 + P_4}{2(1+rv^2)}$
D
D
D
D
P_1
P_2
P_3
P_4
Cash flows:
C

Term Structure Dynamics

- An n-period zero-coupon bond's price can be computed by assigning \$1 to every node at period n and then applying backward induction.
- Repeating this step for $n = 1, 2, \ldots$, one obtains the market discount function implied by the tree.
- The tree therefore determines a term structure.
- It also contains a term structure dynamics.
  - Taking any node in the tree as the current state induces a binomial interest rate tree and, again, a term structure.

Sample Term Structure

- We shall construct interest rate trees consistent with the sample term structure in the following table.
- Assume the short rate volatility is such that $v \equiv r_{\rm h}/r_\ell = 1.5$, independent of time.

Period	1	2	3
Spot rate (%)	4	4.2	4.3
One-period forward rate (%)	4	4.4	4.5
Discount factor	0.96154	0.92101	0.88135

An Approximate Calibration Scheme

- Start with the implied one-period forward rates and then equate the expected short rate with the forward rate (see Exercise 5.6.6 in text).
- For the first period, the forward rate is today's one-period spot rate.
- In general, let f_j denote the forward rate in period j.
- This forward rate can be derived from the market discount function via $f_j = (d(j)/d(j+1)) - 1$ (see Exercise 5.6.3 in text).

An Approximate Calibration Scheme (continued)

- Since the ith short rate $r_j v_j^{i-1}$, $1 \leq i \leq j$, occurs with probability $2^{-(j-1)} \binom{j-1}{i-1}$, this means

$$\sum_{i=1}^{j} 2^{-(j-1)} \binom{j-1}{i-1} r_j v_j^{i-1} = f_j .$$

- Thus

$$r_j = \left(\frac{2}{1+v_j} \right)^{j-1} f_j . \tag{91}$$

- The binomial interest rate tree is trivial to set up.

An Approximate Calibration Scheme (continued)

- The ensuing tree for the sample term structure appears in figure next page.
- For example, the price of the zero-coupon bond paying \$1 at the end of the third period is

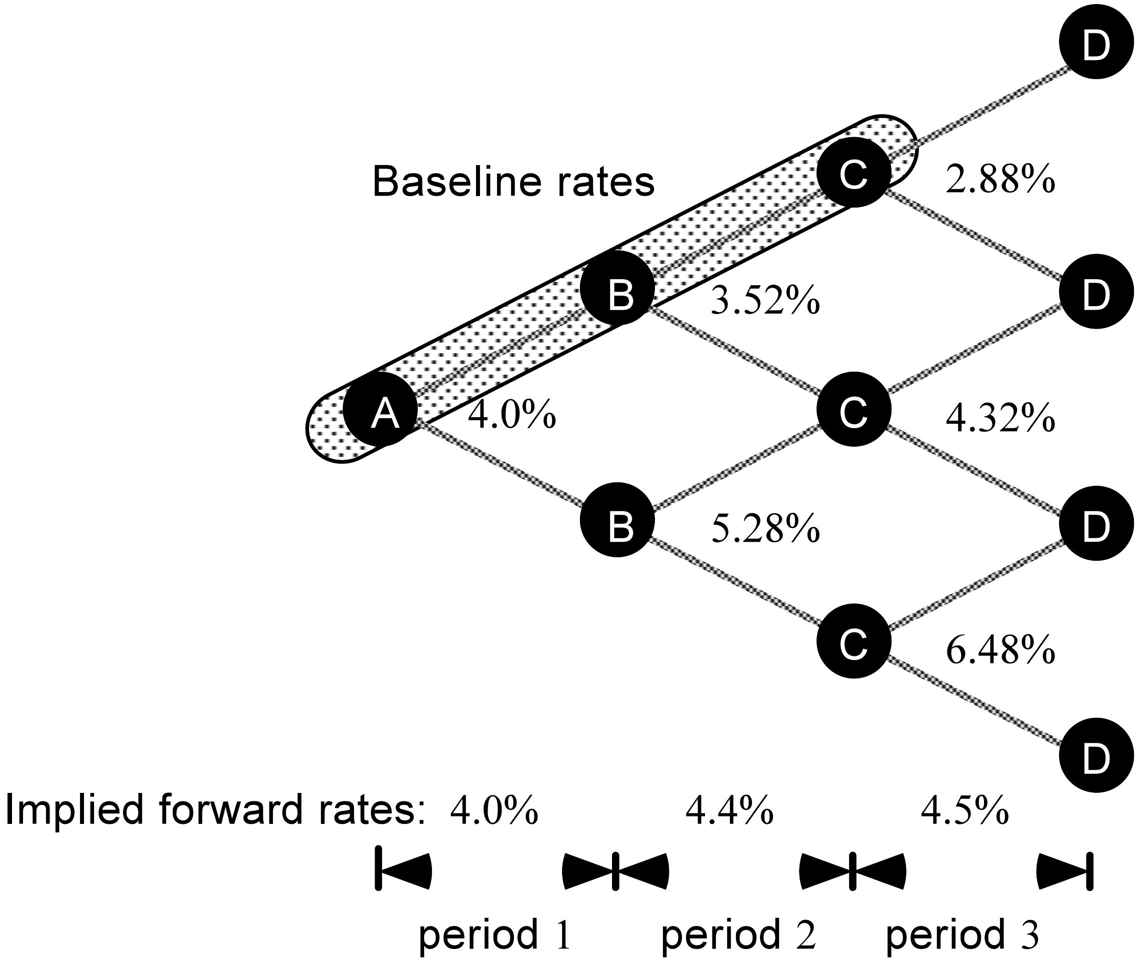
$$\frac{1}{4} \times \frac{1}{1.04} \times \left(\frac{1}{1.0352} \times \left(\frac{1}{1.0288} + \frac{1}{1.0432} \right) + \frac{1}{1.0528} \times \left(\frac{1}{1.0432} + \frac{1}{1.0648} \right) \right)$$

  or 0.88155, which exceeds discount factor 0.88135.
- The tree is thus not calibrated.

An Approximate Calibration Scheme (concluded)

- Indeed, this bias is inherent: The tree overprices the bonds (see text).
- Suppose we replace the baseline rates r_j by $r_j v_j$.
- Then the resulting tree underprices the bonds.[a]
- The baseline rates are thus bounded from above and below.

[a] Lyuu and Wang (`F95922018`) (2009).


D
Baseline rates
C
2.88%
B
3.52%
D
A
4.0%
C
4.32%
B
5.28%
D
C
6.48%
D
Implied forward rates: 4.0%
4.4%
4.5%
period 1
period 2
period 3

Issues in Calibration

- The model prices generated by the binomial interest rate tree should match the observed market prices.
- Perhaps the most crucial aspect of model building.
- Treat the backward induction for the model price of the m-period zero-coupon bond as computing some function of the unknown baseline rate r_m called $f(r_m)$.
- A root-finding method is applied to solve $f(r_m) = P$ for r_m given the zero's price P and $r_1, r_2, \ldots, r_{m-1}$.
- This procedure is carried out for $m = 1, 2, \ldots, n$.
- It runs in cubic time, hopelessly slow.

Binomial Interest Rate Tree Calibration

- Calibration can be accomplished in quadratic time by the use of forward induction.[a]
- The scheme records how much \$1 at a node contributes to the model price.
- This number is called the state price.
  - It is the price of a state contingent claim that pays \$1 at that particular node (state) and 0 elsewhere.
- The column of state prices will be established by moving *forward* from time 1 to time n.

[a] Jamshidian (1991).

Binomial Interest Rate Tree Calibration (continued)

- Suppose we are at time j and there are $j+1$ nodes.
  - The baseline rate for period j is $r \equiv r_j$.
  - The multiplicative ratio be $v \equiv v_j$.
  - $P_1, P_2, \ldots, P_j$ are the state prices at time $j-1$, corresponding to rates $r, rv, \ldots, rv^{j-1}$.
- By definition, $\sum_{i=1}^{j} P_i$ is the price of the $(j-1)$-period zero-coupon bond.
- We want to find r based on $P_1, P_2, \ldots, P_j$ and the price of the j-period zero-coupon bond.

Binomial Interest Rate Tree Calibration (continued)

- One dollar at time j has a known market value of $1/[\,1+S(j)\,]^j$, where $S(j)$ is the j-period spot rate.
- Alternatively, this dollar has a present value of

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

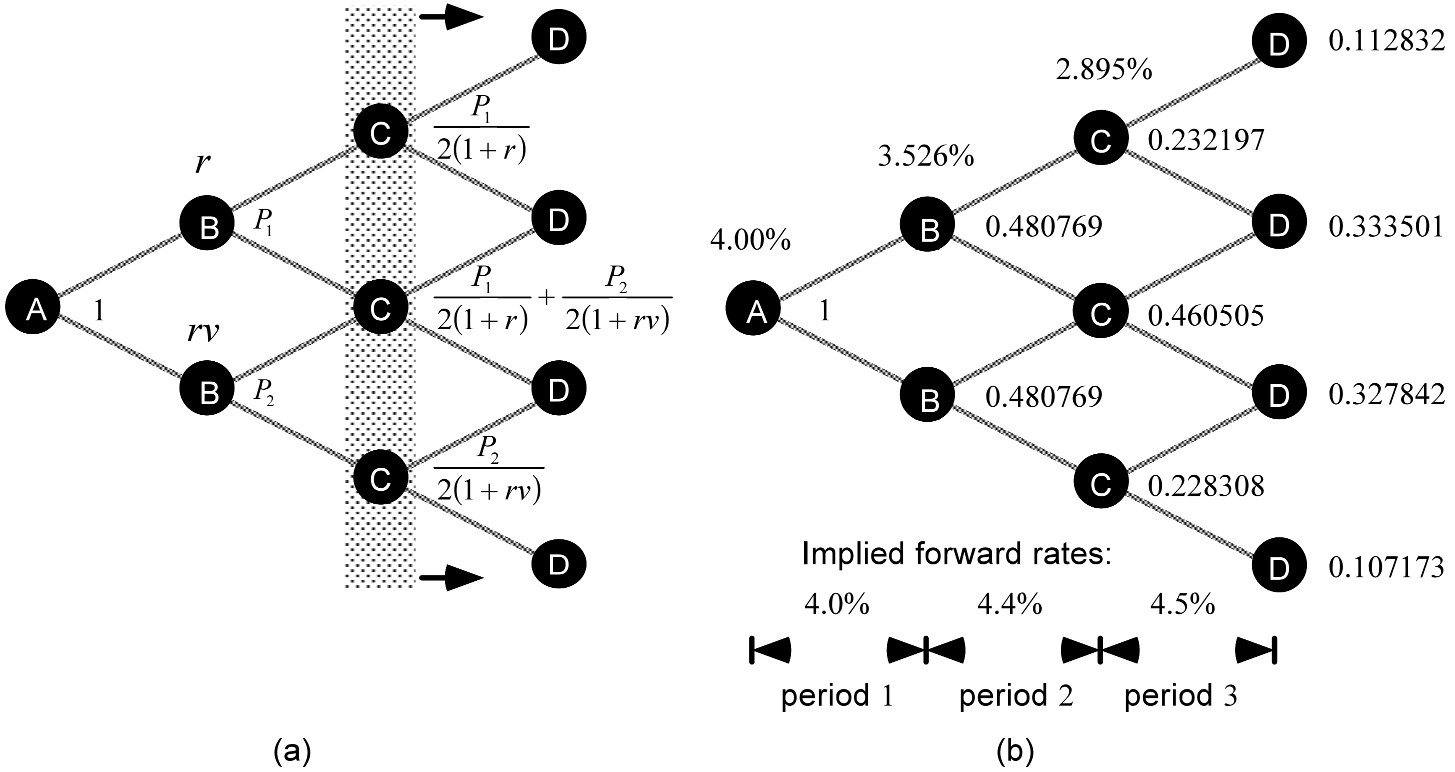
- So we solve

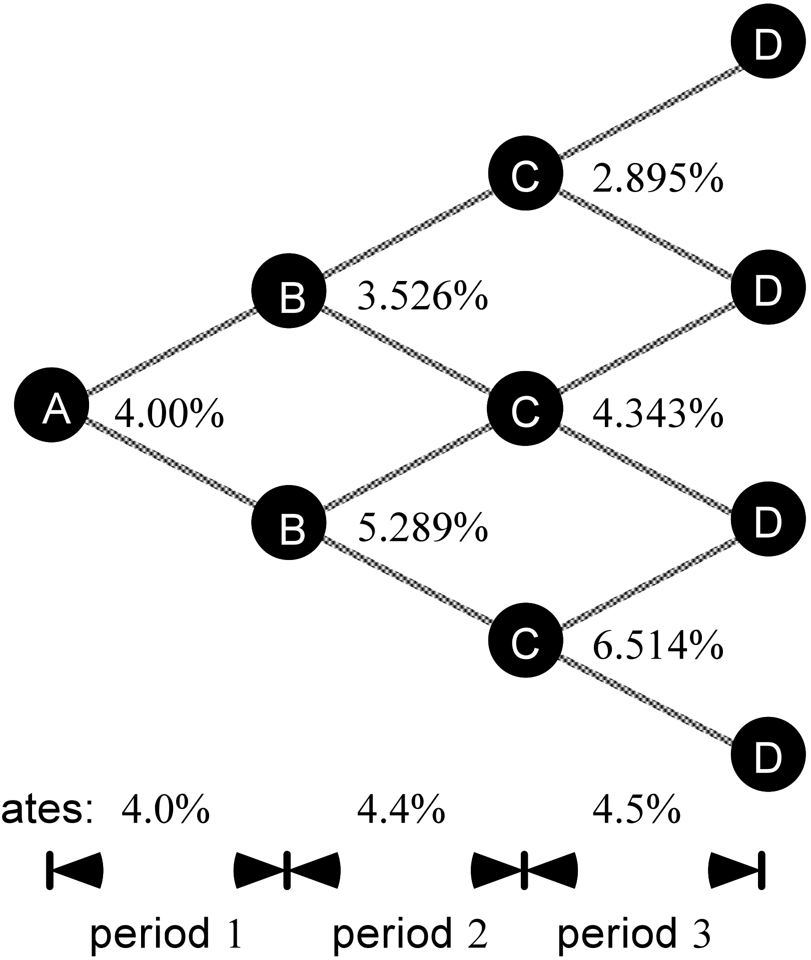
$$g(r) = \frac{1}{[\,1+S(j)\,]^j} \tag{92}$$

for r.

Binomial Interest Rate Tree Calibration (continued)

- Given a decreasing market discount function, a unique positive solution for r is guaranteed.
- The state prices at time j can now be calculated (see figure (a) next page).
- We call a tree with these state prices a binomial state price tree (see figure (b) next page).
- The calibrated tree is depicted on p. 834.


A
B
B
C
C
C
D
D
D
D
1
r
rv
P_1
P_2
$\frac{P_1}{2(1+r)}$
$\frac{P_1}{2(1+r)}+\frac{P_2}{2(1+rv)}$
$\frac{P_2}{2(1+rv)}$
(a)
A
B
B
C
C
C
D
D
D
D
4.00%
1
3.526%
0.480769
0.480769
2.895%
0.232197
0.460505
0.228308
0.112832
0.333501
0.327842
0.107173
Implied forward rates:
4.0%
4.4%
4.5%
period 1
period 2
period 3
(b)


D
C
2.895%
D
B
3.526%
A
4.00%
C
4.343%
B
5.289%
D
C
6.514%
D
Implied forward rates: 4.0% 4.4% 4.5%
period 1
period 2
period 3

Binomial Interest Rate Tree Calibration (concluded)

- The Newton-Raphson method can be used to solve for the r in Eq. (92) on p. 831 as $g'(r)$ is easy to evaluate.
- The monotonicity and the convexity of $g(r)$ also facilitate root finding.
- The total running time is $O(\mathcal{C}n^2)$, where $\mathcal{C}$ is the maximum number of times the root-finding routine iterates, each consuming $O(n)$ work.
- With a good initial guess, the Newton-Raphson method converges in only a few steps.[a]

[a] Lyuu (1999).

A Numerical Example

- One dollar at the end of the second period should have a present value of 0.92101 by the sample term structure.
- The baseline rate for the second period, r_2, satisfies

$$\frac{0.480769}{1+r_2}+\frac{0.480769}{1+1.5\times r_2}=0.92101.$$

- The result is $r_2=3.526\%$.
- This is used to derive the next column of state prices shown in figure (b) on p. 833 as 0.232197, 0.460505, and 0.228308.
- Their sum gives the correct market discount factor 0.92101.

A Numerical Example (concluded)

- The baseline rate for the third period, r_3, satisfies

$$\frac{0.232197}{1+r_3}+\frac{0.460505}{1+1.5\times r_3}+\frac{0.228308}{1+(1.5)^2\times r_3}=0.88135.$$

- The result is $r_3 = 2.895\%$.
- Now, redo the calculation on p. 825 using the new rates:

$$\frac{1}{4}\times\frac{1}{1.04}\times[\frac{1}{1.03526}\times(\frac{1}{1.02895}+\frac{1}{1.04343})+\frac{1}{1.05289}\times(\frac{1}{1.04343}+\frac{1}{1.06514})],$$

  which equals 0.88135, an exact match.

- The tree on p. 834 prices without bias the benchmark securities.

Spread of Nonbenchmark Bonds

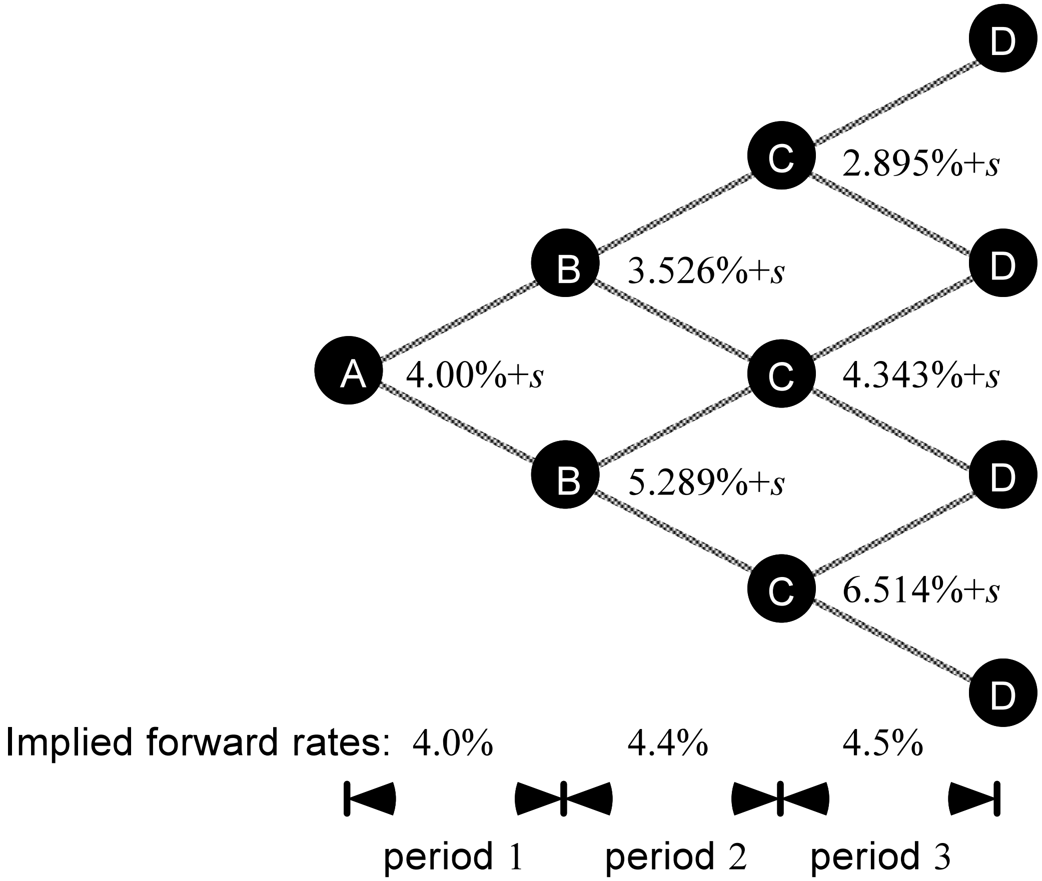
- Model prices calculated by the calibrated tree as a rule do not match market prices of nonbenchmark bonds.
- The incremental return over the benchmark bonds is called spread.
- If we add the spread uniformly over the short rates in the tree, the model price will equal the market price.
- We will apply the spread concept to option-free bonds next.

Spread of Nonbenchmark Bonds (continued)

- We illustrate the idea with an example.
- Start with the tree on p. 840.
- Consider a security with cash flow C_i at time i for $i = 1, 2, 3$.
- Its model price is $p(s)$, which is equal to

$$\begin{aligned} &\frac{1}{1.04+s} \times \left[C_1 + \frac{1}{2} \times \frac{1}{1.03526+s} \times \left(C_2 + \frac{1}{2} \left(\frac{C_3}{1.02895+s} + \frac{C_3}{1.04343+s} \right) \right) + \right. \\ &\qquad \left. \frac{1}{2} \times \frac{1}{1.05289+s} \times \left(C_2 + \frac{1}{2} \left(\frac{C_3}{1.04343+s} + \frac{C_3}{1.06514+s} \right) \right) \right]. \end{aligned}$$

- Given a market price of P, the spread is the s that solves $P = p(s)$.



A

4.00%+*s*

B

3.526%+*s*

B

5.289%+*s*

C

2.895%+*s*

C

4.343%+*s*

C

6.514%+*s*

D

D

D

D

Implied forward rates: 4.0% 4.4% 4.5%

period 1 period 2 period 3

Spread of Nonbenchmark Bonds (continued)

- The model price $p(s)$ is a monotonically decreasing, convex function of s.
- We will employ the Newton-Raphson root-finding method to solve $p(s) - P = 0$ for s.
- But a quick look at the equation for $p(s)$ reveals that evaluating $p'(s)$ directly is infeasible.
- Fortunately, the tree can be used to evaluate both $p(s)$ and $p'(s)$ during backward induction.

Spread of Nonbenchmark Bonds (continued)

- Consider an arbitrary node A in the tree associated with the short rate r.
- In the process of computing the model price $p(s)$, a price $p_{\mathrm{A}}(s)$ is computed at A.
- Prices computed at A's two successor nodes B and C are discounted by $r + s$ to obtain $p_{\mathrm{A}}(s)$ as follows,

$$p_{\mathrm{A}}(s) = c + \frac{p_{\mathrm{B}}(s) + p_{\mathrm{C}}(s)}{2(1 + r + s)},$$

  where c denotes the cash flow at A.

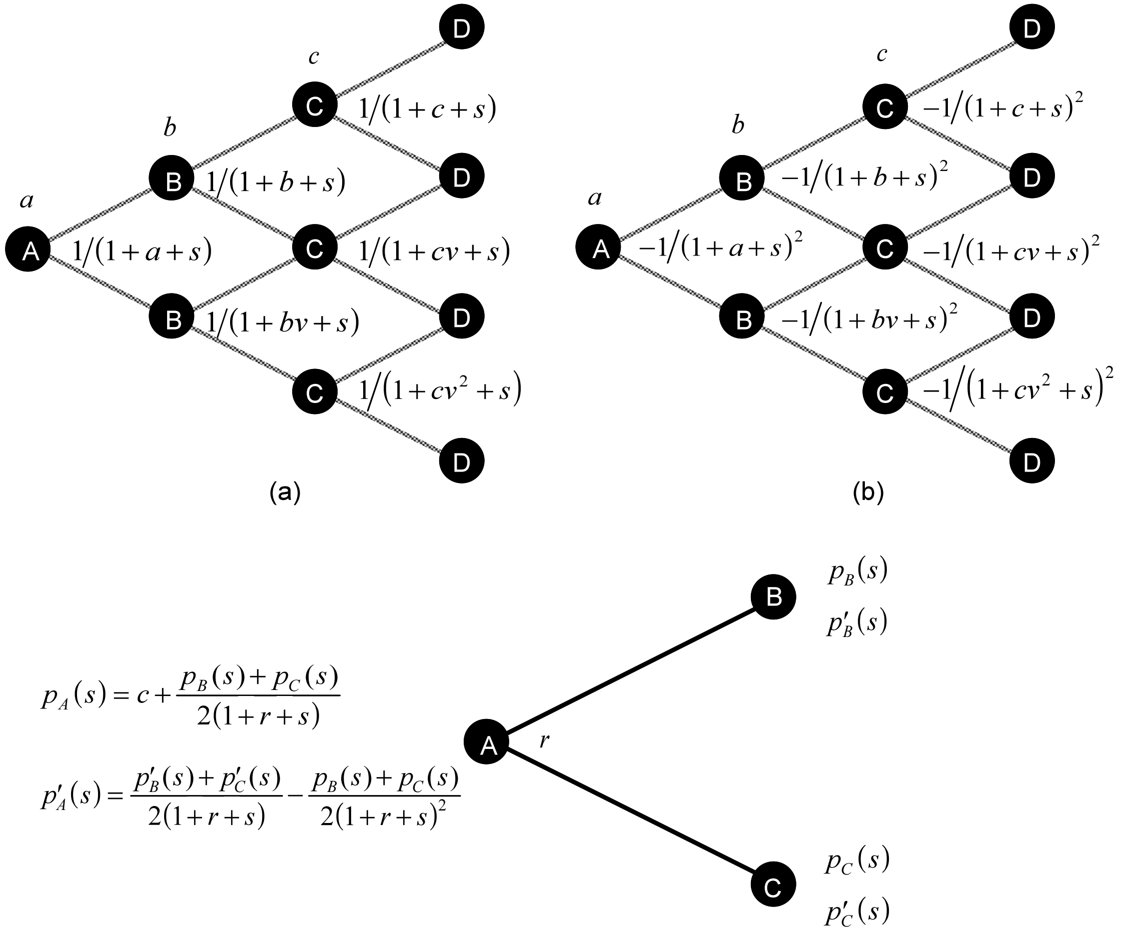
Spread of Nonbenchmark Bonds (continued)

- To compute $p'_{\mathrm{A}}(s)$ as well, node A calculates

$$p'_{\mathrm{A}}(s) = \frac{p'_{\mathrm{B}}(s) + p'_{\mathrm{C}}(s)}{2(1+r+s)} - \frac{p_{\mathrm{B}}(s) + p_{\mathrm{C}}(s)}{2(1+r+s)^2}. \qquad (93)$$

- This is easy if $p'_{\mathrm{B}}(s)$ and $p'_{\mathrm{C}}(s)$ are also computed at nodes B and C.
- Apply the above procedure inductively to yield $p(s)$ and $p'(s)$ at the root (p. 844).
- This is called the differential tree method.[a]

[a] Lyuu (1999).



(a)

(b)

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

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

(c)

Spread of Nonbenchmark Bonds (continued)

- Let $\mathcal{C}$ represent the number of times the tree is traversed, which takes $O(n^2)$ time.
- The total running time is $O(\mathcal{C}n^2)$.
- In practice $\mathcal{C}$ is a small constant.
- The memory requirement is $O(n)$.

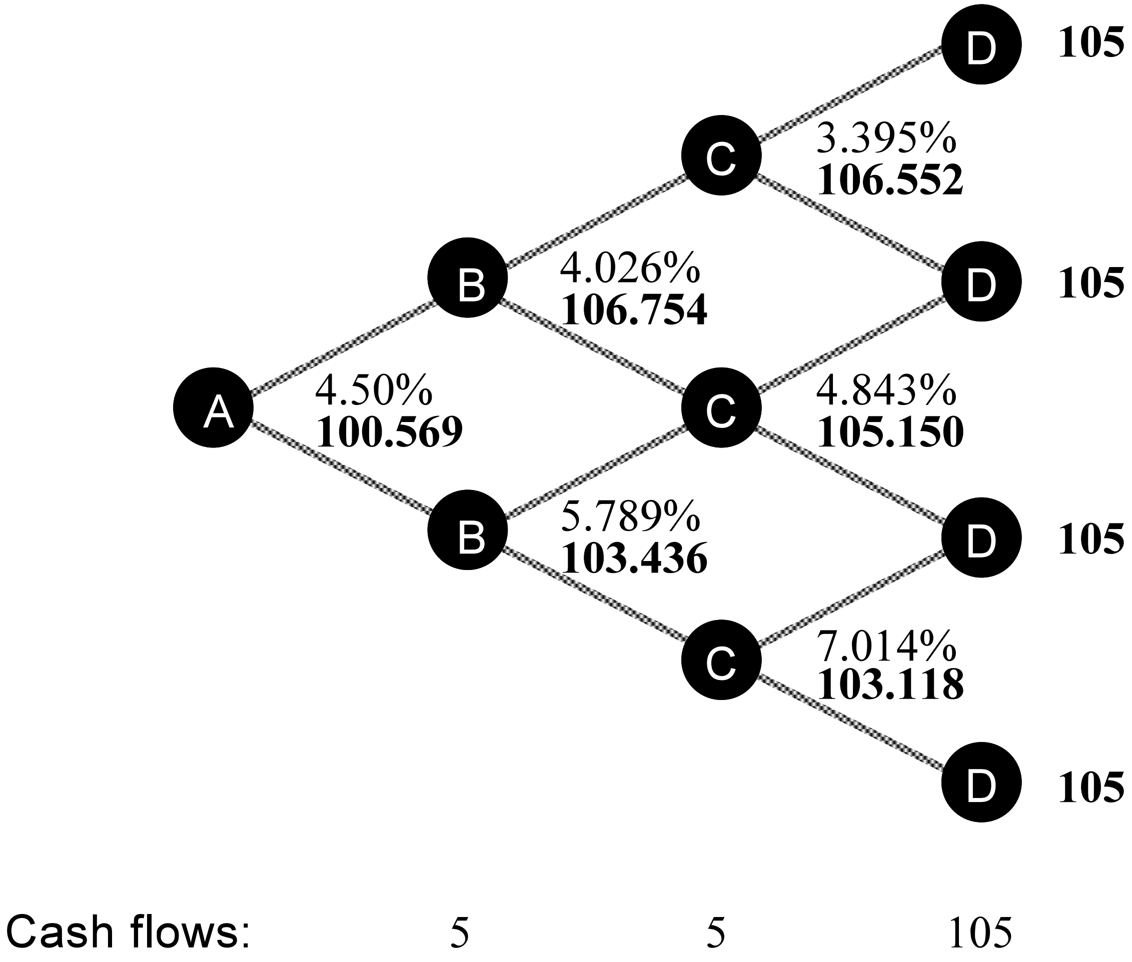
Spread of Nonbenchmark Bonds (continued)

Number of partitions n	Running time (s)	Number of iterations	Number of partitions	Running time (s)	Number of iterations
500	7.850	5	10500	3503.410	5
1500	71.650	5	11500	4169.570	5
2500	198.770	5	12500	4912.680	5
3500	387.460	5	13500	5714.440	5
4500	641.400	5	14500	6589.360	5
5500	951.800	5	15500	7548.760	5
6500	1327.900	5	16500	8502.950	5
7500	1761.110	5	17500	9523.900	5
8500	2269.750	5	18500	10617.370	5
9500	2834.170	5	....	....	....

75MHz Sun SPARCstation 20.

Spread of Nonbenchmark Bonds (concluded)

- Consider a three-year, 5% bond with a market price of 100.569.
- Assume the bond pays annual interest.
- The spread can be shown to be 50 basis points over the tree (p. 848).
- Note that the idea of spread does not assume parallel shifts in the term structure.
- It also differs from the yield spread (p. 113) and static spread (p. 114) of the nonbenchmark bond over an otherwise identical benchmark bond.


D
105
C
3.395%
106.552
B
4.026%
106.754
D
105
A
4.50%
100.569
C
4.843%
105.150
B
5.789%
103.436
D
105
C
7.014%
103.118
D
105
Cash flows:
5
5
105

More Applications of the Differential Tree: Calibrating Black-Derman-Toy (in seconds)

Number	Running	Number	Running	Number	Running
of years	time	of years	time	of years	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

75MHz Sun SPARCstation 20, one period per year.

More Applications of the Differential Tree: Calculating Implied Volatility (in seconds)

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

Intel 166MHz Pentium, running on Microsoft Windows 95.