Black-Derman-Toy: a simple implementation in C

Eric Bouyé

Autumn 1998
Financial Econometrics Research Center
City University Business School
http://www.city.ac.uk/cubs/ferc/eric
e.bouye@city.ac.uk

1 Black-Derman-Toy

1.1 The Model

In the Black-Derman-Toy model, the tree structure is such that the probability of an upward or downward change in the interest rate is the same (0.5).

$$\begin{array}{ccc} & & r_{j+1}^{(t+1)}(\tau) \\ & \overset{0.5}{\nearrow} & \\ r_j^{(t)}(\tau) & & \\ & \underset{0.5}{\searrow} & \\ & & r_j^{(t+1)}(\tau) \end{array}$$

Condition 1 *The binomial tree* ***must*** *reflects the current term structure. Then the following holds:*

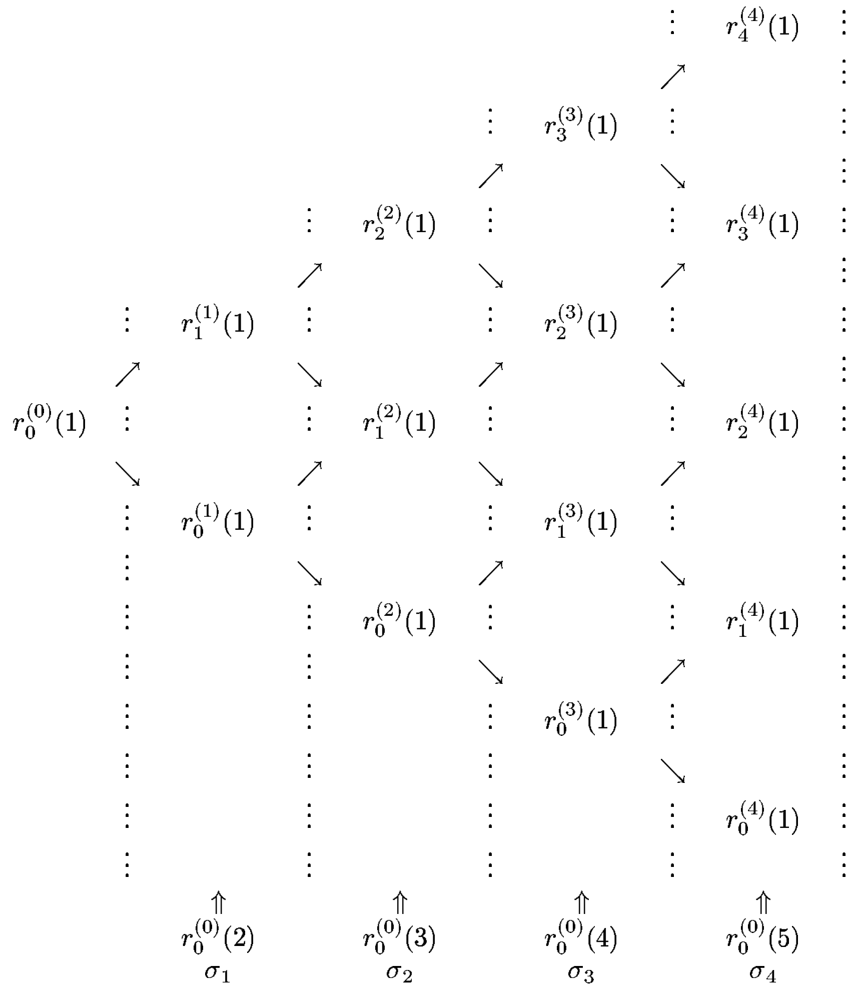
$$\text{for } \tau = 1, ..., n$$

$$\left\{ \begin{array}{ll} g(r_j^{(t)}(1)) = & \left\{ \begin{array}{ll} 1 & \text{if } t = \tau \\ \frac{1}{1+r_j^{(t)}(1)} \left[g(r_{j+1}^{(t+1)}(1)) + g(r_j^{(t+1)}(1)) \right] * 0.5 & \text{otherwise} \end{array} \right. \\ \frac{1}{(1+r_0^{(0)}(\tau))^{\tau}} = & g(r_0^{(0)}(1)) \end{array} \right. \tag{1}$$

However, this condition is not sufficient to have a closed-form solution. Benninga and Wiener [1997] introduce a relationship such that we can obtain a closed-form solution to this non-linear problem.

Condition 2 *There exists a relationship between the interest rates of the same period t.*

$$r_j^{(t)}(1) = r_0^{(t)}(1) \exp(2j\sigma_t) \quad \text{for } j = 1, ..., t. \tag{2}$$



2 The program

There are two problems to solve. First, the recursive function $g(r_j^{(t)}(1))$ corresponding to the first expression of equation (2.1) has to be implemented. The function builds all possible ways in the tree. It is the main difference with Ho-Lee model. Second, a bisection algorithm finds a root for the non-linear equation in (2.1) taking into account volatilities from (2.2). This choice is motivated for practical purpose and simplicity. Indeed, it would have been possible to use Brent or Newton-Raphson algorithms to increase the rate of convergence.

Algorithm 3 *treebuild(t,j) {*
if $t < n$ { $r_j^{(t)}(1) \longleftarrow r_0^{(t)}(1)\exp(2j\sigma_t)$
return $\frac{1}{1+r_j^{(t)}(1)}$*(treebuild(t+1,j+1)+treebuild(t+1,j))* }
else return 1.0 }
treebuild(0,0)(0.5)^n*

The C code for the Black-Derman-Toy model is:

```
#include <stdio.h>
#include <math.h>

#define NTAU 5
#define DIMTREE 15
```

Simple bisection algorithm

```
double bisection(double (*f)(double[], double[], int, double),
                 double a,
                 double b,
                 double rmin[],
                 double vol[],
                 int n,
                 double R)
{
```

The searched element of the array rmin is allocated with the bounds

```
  double fa,fb,c,t,side;
  rmin[n-1] = b;
  fb = (*f)(rmin,vol,n,R);
  rmin[n-1] = a;
  fa = (*f)(rmin,vol,n,R);
  if (fb*fa > 0) {
    printf(''Change DELTA value for convergence\n'');
    return 0.0;
  }
  else {
    if (fb >= fa) side = 1.0; else side = 0.0;
    while (fabs(b-a)>1e-8) {
      c = (a+b)/2.0;
    rmin[n-1] = c;
      if ( (*f)(rmin,vol,n,R) > 0) t = 1; else  t = 0;
      a = side*(t*a + (1-t)*c) + (1-side)*(t*c + (1-t)*a);
      b = side*(t*c + (1-t)*b) + (1-side)*(t*b + (1-t)*c);
    }
    c = (a+b)/2.0;
    return c;
  }
}
```

Equation (2.2)

```
double BenningaWiener(int t, int j, double rmin[], double vol[])
{
  return rmin[t] * exp(2*vol[t]*j);
}
```

The main piece of the program: the recursive function

```
double treebuild(int t, int j, double rmin[], double vol[], int n)
{
  double y, rtj;

  if ( t < n ) {
    rtj = BenningaWiener(t,j,rmin,vol);
    y = (1/(1+rtj))*(treebuild(t+1,j+1,rmin,vol,n) + treebuild(t+1,j,rmin,vol,n));
 }
 else y =1.0;
 return y;
}
```

Function used for bisection

```
double function(double rmin[], double vol[], int n, double R)
```

```
{
  return (R-treebuild(0,0,rmin,vol,n)/pow(2,n));
}
```

This function defines the bounds and allocate node values

```
void BDT(double tree[], double zerorate[], double volatility[], double delta)
{
  int i,indx,n;
  double a,b,R;
  double rmin[NTAU];

  rmin[0] = zerorate[1];
  tree[0] = rmin[0];
  indx = 0;
  for(n=1;n<NTAU;n++) {
    a = rmin[n-1] - delta;
    b = rmin[n-1] + delta;
    R = 1/pow((1+zerorate[n+1]),n+1);
    rmin[n] = bisection(&function,a,b,rmin,volatility,n+1,R);
    indx += n;
    tree[indx] = rmin[n];
    for(i=1;i<=n;i++) tree[indx+i] = BenningaWiener(n,i,rmin,volatility);
 }
}
```

The famous example of Black, Derman and Toy [1990]

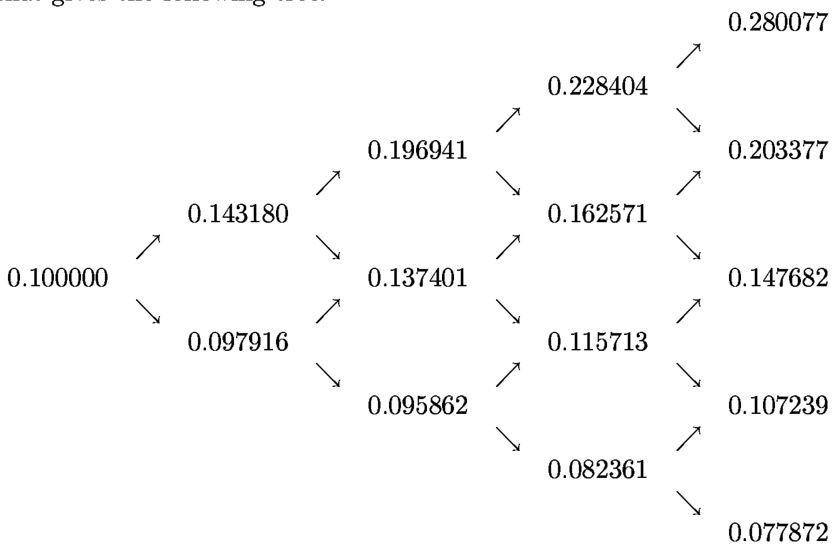
```
int main()
{
  int i;
  double Delta;
  double Tree[DIMTREE+1]; double ZeroRate[NTAU+1]; double Volatility[NTAU];
  ZeroRate[1] = 0.10;
  ZeroRate[2] = 0.11;
  ZeroRate[3] = 0.12;
  ZeroRate[4] = 0.125 ;
  ZeroRate[5] = 0.13;
  Volatility[0] = 0.00;
  Volatility[1] = 0.19 ;
  Volatility[2] = 0.18 ;
  Volatility[3] = 0.17 ;
  Volatility[4] = 0.16;
  Delta = 0.02;

  BDT(Tree,ZeroRate,Volatility,Delta);
  for(i=0;i<DIMTREE;i++) printf(''%lf\n'',Tree[i]);
  return 0;
}
```

The output of this program is:

```
0.100000
0.097916
0.143180
0.095862
0.137401
0.196941
0.082361
```

```
0.115713
0.162571
0.228404
0.077872
0.107239
0.147682
0.203377
0.280077
```

that gives the following tree:



References

[1] BENNINGA S. AND Z. WIENER [1998], Binomial term structure models, *Lecture Notes*.

[2] BLACK F., E. DERMAN AND W. TOY [1990], A one-factor model of interest rates and its application to treasury bond option, Financial Analysts Journal, **46**, 33-39.

[3] BLACK F. AND M. SCHOLES [1973], The pricing of options and corporate liabilities, *Journal of Political Economy*, **81**, 637-59.

[4] CHALASANI P. AND S. JHA [1997], Steven Shreve: Stochastic Calculus and Finance, *Postscript*, Department of Mathematical Sciences, Carnegie Mellon University.

[5] EL KARAOUI N. [1996], Modèles stochastiques en Finance, *Lecture Notes*, Université Paris VI.

[6] MERMET B. [1996], C, Algorithmique et Programmation, *Lecture Notes*, DEA de Chimie Informatique et Theorique.

[7] RONCALLI T. [1998], La structure par terme des taux zéros: modélisation et implémentation numérique, *Thèse ès Sciences Economiques*.