CHAPTER 7

Binomial Option Pricing Model

7.1. Risk-Neutral Option Pricing and Binomial Tree

Options are financial instruments that convey the right, but not the obligation, to enter into a future transaction on an underlying asset. In stochastic model, asset price return during the time increment from t to $t + \Delta t$ is assumed to follow a random normal process as

$$\Delta S_t / S_t = \varepsilon(\mu \Delta t , \sigma \sqrt{\Delta t}) \tag{7.1}$$

where μ and σ are respectively the mean rate and volatility of return. For constant and flat interest rate r, current price of an option written on this asset can be defined based on the present value of its average maturity payoff at time T as[1]

$$f_0 = e^{-rT} \hat{E}(f_T \mid S_0) \tag{7.2}$$

In (7.2), we are averaging over realized maturity payoffs of the option f_T in respect to sample asset prices generated through the so-called risk-neutral process related to (7.1). The option price f_0 is said to be evaluated at current asset price S_0 that initiates the risk-neutral process. Equation (7.2) is referred as the risk-neutral option pricing that proven to be equivalent to the Black-Scholes differential equation. For traded underlying asset such as stock, it can be shown that the risk-neutral process is simply given by (7.1) with the replacement of the mean rate μ by the interest rate r. The risk-neutral average in (7.2) can only be calculated analytically for options with simple structure. In general, it would be highly intense for options with exotic exercising condition.

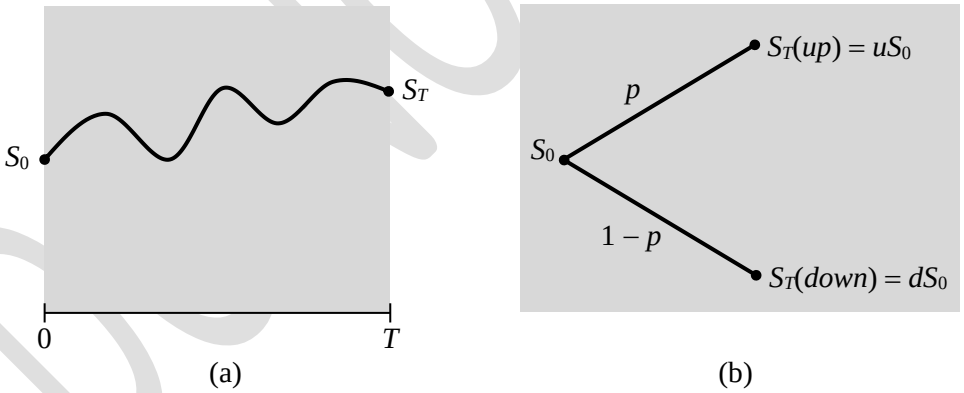


Figure 7.1: (a) Risk-neutral process of asset price, and (b) its one-step binomial tree representation.

In Figure 7.1(a), the risk-neutral process of asset price starts off from S_0 and ends up with S_T at option's maturity. The k-th statistical moment of the maturity price under the risk-neutral process is given by

$$\hat{E}(S_T^k) = S_0^k \, e^{\, krT + \frac{1}{2}k(k-1)\sigma^2 T} \tag{7.3}$$

[1] Denote option price $f_t = f(S_t , t)$ at time t. It can be shown using Ito's lemma that $e^{-rt} f_t$ is a martingale under risk-neutral preference. This gives $e^{-rt} f_t = \hat{E}(e^{-rs} f_s \mid S_t)$ for $s > t$, and the conditional expectation is evaluated based on the risk-neutral process of the underlying asset price starting off from S_t.

In binomial model, we can simplify the calculation of the risk-neutral average in (7.2) by adopting a binomial representation for the price movement in Figure 7.1(a). The simplest model is the one-step binomial tree initiated from S_0 with either up or down scenario for the maturity price as shown in Figure 7.1(b). In general, it requires three factors in the parameterization namely the branching probability p, the up factor u, and the down factor d. They can be determined by matching the statistical moments of the maturity price as in (7.4) such that the binomial step will mimic the leading statistical properties of a full stochastic description.

$$p(uS_0)^k + (1-p)(dS_0)^k = S_0^k\, e^{\,krT + \frac{1}{2}k(k-1)\sigma^2 T} \tag{7.4}$$

In this way, the risk-neutral average in (7.2) can be estimated to be

$$\hat{E}(\, f_T\,) = p\, f(\, uS_0\,,\, T\,) + (1-p)\, f(\, dS_0\,,\, T\,) \tag{7.5}$$

It should be noted that the risk-neutral process itself is parameterized by only two factors r and σ in the stochastic model. It is therefore not admissible to find three factors $\{\, p, u, d\,\}$ that satisfy the matching condition (7.4) simultaneously for k equals 1, 2, and 3.

In Cox-Ross-Rubinstein parameterization[2], we define instead the binomial tree in Figure 7.1(b) with two factors $\{\, p, u, d = 1/u\,\}$ and match (7.4) simultaneously for the first two moments. This gives

$$p = \frac{e^{\,rT} - (1/u)}{u - (1/u)} \tag{7.6}$$

$$u = \tfrac{1}{2}(\, e^{-rT} + e^{\,(r+\sigma^2)T}\,) + \tfrac{1}{2}\sqrt{(\, e^{-rT} + e^{\,(r+\sigma^2)T}\,)^2 - 4} \tag{7.7}$$

For the higher-order moments ($k \geq 3$) in (7.4), it can be shown that the discrepancies are always in the second order of the stochastic factors rT and $\sigma^2 T$ using the parameters p and u as defined above[3]. Thus, the error involved on the estimation in (7.5) will be insignificant if both the sizes of rT and $\sigma^2 T$ are small compared with one. Alternatively, in a more symmetric parameterization by Jarrow and Rudd[4] defined as $\{\, p = \tfrac{1}{2}, u, d\,\}$, the binomial factors are calculated in the same way to be

$$u = e^{rT}\,(\, 1 + \sqrt{e^{\sigma^2 T} - 1}\,) \tag{7.8}$$

$$d = e^{rT}\,(\, 1 - \sqrt{e^{\sigma^2 T} - 1}\,) \tag{7.9}$$

In this case, the discrepancies for the higher-order moments are shown to be in the second order of the factor $\sigma^2 T$ only[5].

To improve accuracy and especially when there are intermediate boundary conditions for the option, it is essential to extend the binomial model into multiple steps of n with equal time interval of $\Delta t = T/n$ as depicted in Figure 7.2(a). Consider the particular subtree in Figure 7.2(b) that goes from

---

[2] See J. Cox, S. Ross, and M. Rubinstein, "Option pricing: A simplified approach", Journal of Financial Economics, Vol. 7, Issue 3 (1979) p229-263.

[3] In the first order of rT and $\sigma^2 T$, it can be shown using (7.6) and (7.7) that

$$\Lambda(k) = [\, u^k - (1/u)^k\,]/[\, u - (1/u)\,] \cong k + (1/6)k(k-1)(k+1)\sigma^2 T$$

This gives $p(u)^k + (1-p)(1/u)^k = e^{rT}\Lambda(k) - \Lambda(k-1) \cong 1 + krT + \tfrac{1}{2}k(k-1)\sigma^2 T$.

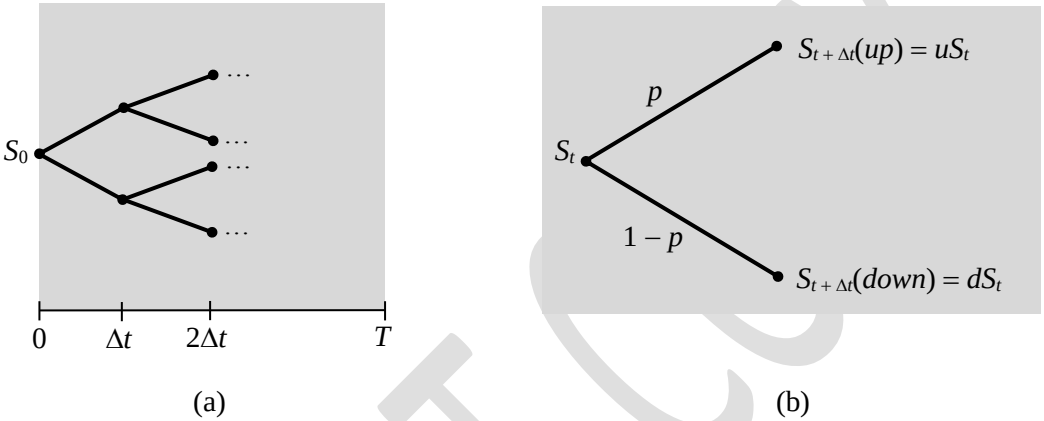
[4] See R.A. Jarrow and A. Rudd, "Option pricing", Richard D. Irwin, Homewood, Illinois (1983).

[5] In this case, we have $p(u)^k + (1-p)(d)^k = \tfrac{1}{2}(\, u^k + d^k\,)$ and it is easy to show that $u^k + d^k \cong e^{krT}\,(\, 2 + k(k-1)\sigma^2 T\,)$ in the first order of $\sigma^2 T$ using (7.8) and (7.9).

time t to $t + \Delta t$ with initial asset price of S_t. Here, the binomial factors can be determined by matching the statistical moments of the end price $S_{t+\Delta t}$ under the risk-neutral process and conditional to the initial price as

$$p(uS_t)^k + (1-p)(dS_t)^k = S_t^k \, e^{kr\Delta t + \frac{1}{2}k(k-1)\sigma^2\Delta t} \tag{7.10}$$

It is obvious from (7.10) that the factors are determined to be independent of the initial price. For constant volatility of return, they are also considered to be universal for every subtree in the multiple-step representation. In Cox-Ross-Rubinstein or Jarrow-Rudd parameterization, the binomial factors are given by equations (7.6) to (7.9) with time interval T replaced by Δt. The discrepancies in higher-order moments are now in the second order of the smaller quantities $r\Delta t$ and $\sigma^2\Delta t$.



(a) (b)

Figure 7.2: (a) Multiple-step binomial tree of asset price. (b) One-step subtree from time t to $t + \Delta t$ with initial asset price S_t.

For an n-step binomial model, there are 2^n ways of how S_0 will evolve into the maturity price and pick up certain sequence of u and d factors along the tree. To evaluate the risk-neutral average in (7.2), we need to keep track of the option's payoff for each of these scenarios in respect of the exercising conditions. This renders the calculation to be highly inefficient for very large n. Since the u and d factors are universal for every subtree, tree nodes are recombining in the way that an up movement followed by a down will have the same asset price as in its reverse order. As shown in Figure 7.3, this makes the number of end nodes to grow like $n + 1$ and the total number of nodes for the entire n-step tree is manageable at $\frac{1}{2}(n + 1)(n + 2)$ a lot less than the scenarios.

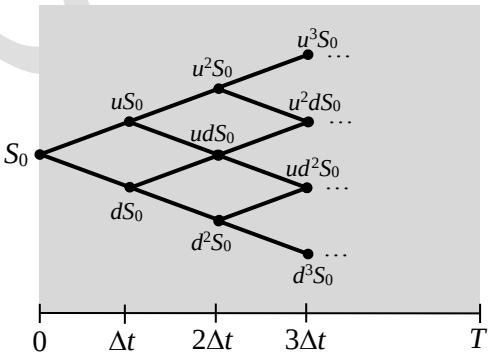


Figure 7.3: Multiple-step binomial tree with recombining nodes.

At time $t = i\Delta t$, there are $i + 1$ nodes from top to bottom of the tree with asset prices defined to be $S_{ij} = u^{i-j}d^j S_0$ where j runs from 0 to i. The j-th node is connected to both the j-th and $(j + 1)$-th nodes

at later time $t + \Delta t$ through a subtree. It is then efficient to consider instead an iterative form of risk-neutral pricing with respect to all these subtrees as

$$f(S_{ij} , t) = e^{-r\Delta t} [p\, f(S_{i+1\,j} , t + \Delta t) + (1 - p)\, f(S_{i+1\,j+1} , t + \Delta t)] \ , \quad j = 0, \ldots , i \qquad (7.11)$$

Equation (7.11) allows us to generate the option prices at time t based on the option prices at later time $t + \Delta t$. At option maturity $T = n\Delta t$, there are $n + 1$ nodes with asset prices $S_{nj} = u^{n-j}d^j S_0$ where j runs from 0 to n. We can start off the iteration from the maturity payoffs $\psi(S_T)$ of the option and work backward in time toward its current value $f(S_0 , 0)$. For exotic options with intermediate boundary conditions, we need to adjust the risk-neutral pricings in (7.11) according to the boundary conditions before the next iteration to an earlier time. For example, an American style option can be exercised at any time prior to its maturity based on the same payoff function $\psi(S_{ij})$. We should therefore compare each $f(S_{ij} , t)$ in (7.11) with its intrinsic value and perform the update according to the early exercising condition as

$$f(S_{ij} , t) = max\{ f(S_{ij} , t) , \psi(S_{ij}) \} \qquad (7.12)$$

7.2. VBA Implementation

The iteration of option prices in (7.11) can be implemented very easily in VBA. We first develop a routine called GenOptionTree( ) that performs the iteration and generates the option prices at every node in the binomial tree. The pseudo code of GenOptionTree( ) is given by Code 7.1. It requires the input of option parameters (T , K , S_0 , r , σ) and tree configuration (n , *treetype*), where *treetype* specifies the use of Cox-Ross-Rubinstein or Jarrow-Rudd parameterization. The routine returns the array of iterated option prices $f_{ij} = f(S_{ij} , i\Delta t)$ as well as the array of asset prices S_{ij} at every tree node with time label i runs from 0 to n and node label j runs from 0 to i. The iteration starts off from the maturity payoffs of the option at end nodes with $i = n$. The payoff condition is defined through an external function payoff(K, S) with strike price K. It works backward in time from $i = n - 1$ to $i = 0$ and applies (7.11) to the subtree of every node in each column. Intermediate boundary conditions are defined using the routine Boundary(K , S , f) that updates the iterated option price immediately after its risk-neutral generation. The VBA code of GenOptionTree( ) is given by Code 7.2 together with the Payoff( ) function and Boundary( ) routine defined in particular for American put option according to (7.12).

	A	B	C	D	E	F	G	H	I
1									
2	***American Put Option :***								
3									
4	***Maturity (T) =***	1	*(year)*						
5	***Strike (K) =***	50							
6	***Asset Price (S_0) =***	50							
7	***Risk-free Rate (r) =***	0.05	*(per year)*						
8	***Volatility (σ) =***	0.25	*(per year)*						
9									
10	***Binomial Tree Parameterization =***	Cox-Ross-Rubinstein							
11	***Number of Time Steps (n) =***	10							
12	***Display Binomial Tree =***	Yes							
13									
14	***Option Price (f_0) =***	3.959	Binomial Pricing						
15									
16	***Binomial Tree***								
17	***Forward Time(year)***	**0**		**0.1**		**0.2**		**0.3**	
18	***Option & Asset Prices***	3.959	50.000	2.365	54.138	1.197	58.619	0.463	63.470
19				5.670	46.178	3.612	50.000	1.979	54.138
20						7.885	42.649	5.359	46.178
21								10.611	39.389
22									
23									

Figure 7.4: Spreadsheet design of binomial option pricing.

Figure 7.4 depicts the spreadsheet design for this VBA implementation[6]. The button labeled as "Binomial Pricing" will trigger the main VBA routine called BinomialPricing( ) with VBA code given by Code 7.3. The option parameters and tree configuration are inputted into this routine through the named cells B4(maturity), B5(strike), B6(assetprice), B7(riskfree), B8(sigma), B10(treetype), and B11(n). It will call GenOptionTree( ) for the price arrays and the resulting option price of f_{00} will be outputted to cell B14. The entire option trees will be displayed optionally in spreadsheet with respect to the choice of "Yes" in cell B12. As reference, the corresponding asset prices will also be displayed adjacent to the option prices. This can be done by running over both the time and node labels for the price arrays, and allocating cells through row and column offsets from the reference cell B17 for j and i, respectively. To display the option and asset prices in alternative columns, we have adopt even column offset $2i$ for option prices and odd column offset $2i + 1$ for asset prices. The corresponding forward time on the tree will also be displayed along the header row using even column offset from B17. It should be noted that there are all together 255 columns in spreadsheet. The number of time steps in B11 should be taken below 127 when we choose "Yes" in cell B12. It is then necessary to impose a validation check for the cell B11 as

= IF(AND(B12 = "Yes" , B11 > = 127) , FALSE , TRUE)

under **Data**, **Validation**, and **Setting**s with **Allow** chosen to be **Custom** and apply the above condition in **Formula**.

[6] Refer to binomialtree_ap.xls

GenOptionTree(T , K , S_0 , r, σ , n , *treetype* , $S(0:n\ ,\ 0:n)$, $f(0:n\ ,\ 0:n)$)

define the size of the time interval

$\Delta t = T/n$

define the tree factors in Cox-Ross-Rubinstein or Jarrow-Rudd parameterization

If(*treetype* = "Cox-Ross-Rubinstein") then

$u = ½(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t}) + ½\sqrt{(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t})^2 - 4}$, $d = 1/u$, $p = (e^{r\Delta t} - 1/u)/(u - 1/u)$

Elseif(*treetype* = "Jarrow-Rudd") then

$u = e^{r\Delta t} (1 + \sqrt{e^{\sigma^2\Delta t} - 1})$, $d = e^{r\Delta t} (1 - \sqrt{e^{\sigma^2\Delta t} - 1})$, $p = ½$

Endif

setting up the maturity payoffs of the option

For($j = 0$ to n){ $S(n\ ,\ j) = u^{n-j}d^j S_0$

$f(n\ ,\ j) = Payoff(K\ ,\ S(n\ ,\ j))$ }

iterate the option price backward in time and update according to intermediate boundary condition

For($i = n - 1$ to 0){

For($j = 0$ to i){ $S(i\ ,\ j) = u^{i-j}d^j S_0$

$f(i\ ,\ j) = e^{-r\Delta t} [p\, f(i+1\ ,\ j) + (1-p)\, f(i+1\ ,\ j+1)]$

Call *Boundary*(K , $S(i\ ,\ j)$, $f(i\ ,\ j)$) } }

Code 7.1 : Pseudo code of the GenOptionTree( ) routine.

```
Sub GenOptionTree(maturity As Double, strike As Double, assetPrice As Double, riskFree As Double, sigma As Double, _
        n As Integer, treetype As Variant, ByRef STree() As Double, ByRef fTree() As Double)

  Dim St As Double, ft As Double
  Dim u As Double, d As Double, p As Double
  Dim i As Integer, j As Integer
  Dim dtime As Double: dtime = maturity / n

  If (treetype = "Cox-Ross-Rubinstein") Then
    u = 0.5 * (Exp(-riskFree * dtime) + Exp((riskFree + sigma ^ 2) * dtime)) _
      + 0.5 * Sqr((Exp(-riskFree * dtime) + Exp((riskFree + sigma ^ 2) * dtime)) ^ 2 - 4)
    d = 1 / u
    p = (Exp(riskFree * dtime) - 1 / u) / (u - 1 / u)
  ElseIf (treetype = "Jarrow-Rudd") Then
    u = Exp(riskFree * dtime) * (1 + Sqr(Exp(sigma ^ 2 * dtime) - 1))
    d = Exp(riskFree * dtime) * (1 - Sqr(Exp(sigma ^ 2 * dtime) - 1))
    p = 0.5
  End If

  For j = 0 To n
    St = u ^ (n - j) * d ^ (j) * assetPrice
    STree(n, j) = St
    fTree(n, j) = Payoff(strike, St)
  Next j

  For i = n - 1 To 0 Step -1
    For j = 0 To i
      St = u ^ (i - j) * d ^ (j) * assetPrice
      STree(i, j) = St
      ft = Exp(-riskFree * dtime) * (p * fTree(i + 1, j) + (1 - p) * fTree(i + 1, j + 1))
      Call Boundary(strike, St, ft)
      fTree(i, j) = ft
    Next j
  Next i

End Sub
```

```
Function Payoff(strike As Double, assetPrice As Double) As Double
  Payoff = Max(strike - assetPrice, 0)
End Function
```

```
Sub Boundary(strike As Double, assetPrice As Double, optionPrice As Double)
  optionPrice = Max(optionPrice, Payoff(strike, assetPrice))
End Sub
```

```
Function Max(x As Double, y As Double) As Double
  If x > y Then Max = x Else Max = y
End Function
```

Code 7.2 : VBA code of the GenOptionTree( ) routine together with the Payoff( ) function and Boundary( ) routine defined for American put option.

```
Sub BinomialPricing()
  Dim maturity As Double: maturity = Range("maturity").Value
  Dim strike As Double: strike = Range("strike").Value
  Dim riskFree As Double: riskFree = Range("riskFree").Value
  Dim sigma As Double: sigma = Range("sigma").Value
  Dim assetPrice As Double: assetPrice = Range("assetPrice").Value
  Dim n As Integer: n = Range("n").Value
  Dim treetype As Variant: treetype = Range("treetype").Text
  Dim fTree() As Double: ReDim fTree(0 To n, 0 To n)
  Dim STree() As Double: ReDim STree(0 To n, 0 To n)
  Dim i As Integer

  Call GenOptionTree(maturity, strike, assetPrice, riskFree, sigma, n, treetype, STree, fTree)
  Range("B14").Value = fTree(0, 0)

  Range("B17:IV144").ClearContents

  If (Range("B12").Text = "Yes") Then
    For i = 0 To n
      Range("B17").Offset(0, 2 * i) = i * (maturity / n)
      For j = 0 To i
        Range("B17").Offset(j + 1, 2 * i) = fTree(i, j)
        Range("B17").Offset(j + 1, 2 * i + 1) = STree(i, j)
      Next j
    Next i
  End If
End Sub
```

Code 7.3 : VBA code of the main BinomialPricing( ) routine.