Chapter 42
Binomial OPM, Black-Scholes OPM and Their Relationship: Decision Tree and Microsoft Excel Approach

John Lee

Abstract This paper will first demonstrate how Microsoft Excel can be used to create the Decision Trees for the Binomial Option Pricing Model. At the same time, this paper will discuss the Binomial Option Pricing Model in a less mathematical fashion. All the mathematical calculations will be done by the Microsoft Excel program that is presented in this paper. Finally, this paper uses the Decision Tree approach to demonstrate the relationship between the Binomial Option Pricing Model and the Black-Scholes Option Pricing Model.

Keywords Binomial Option Pricing Model • Decision trees • Black-Scholes Option Pricing Model • Call options • Put options • Microsoft Excel • Visual basic for applications (VBA) • Put-call parity • Sigma • Volatility • Recursive programming

42.1 Introduction

The Binomial Option Pricing Model derived by Rendleman and Barter (1979), and Cox et al. (1979) is one the most famous models used to price options. Only the Black-Scholes model (1973) is more famous. One problem with learning the Binomial Option Pricing Model is that it is computationally intensive, which makes it a very complicated formula.

The complexity of the Binomial Option Pricing Model makes it a challenge to learn the model. Most books teach the Binomial Option model by describing the formula. This is a not very effective because it usually requires the learner to mentally keep track of many details, many times to the point of information overload. There is a well-known principle in psychology that the average number of things that a person can remember at one time is seven.

This paper will first demonstrate the power of Microsoft Excel. It will do this by demonstrating that it is possible to create large decision trees for the Binomial Pricing Model using Microsoft Excel. A ten-period decision tree would require 2,047 call calculations and 2,047 put calculations. This paper will also show the decision tree for pricing a stock and a bond, each requiring 2,047 calculations. Therefore, there would be $2,047 \times 4 = 8,188$ calculations for a complete set of ten-period decision trees.

Second, this paper will present the Binomial Option Model in a less mathematical fashions. It will try to make it so that the reader will not have to keep track of many things at one time. It will do this by using decision trees to price call and put options.

Finally, we will show the relationship between the Binomial Option Pricing Model and the Black-Scholes Option Pricing Model.

This paper uses a Microsoft Excel workbook called *binomialBS_OPM.xls* that contains the VBA code to create the decision trees for the Binomial Option Pricing Model. The VBA code is published in Appendix 42A. E-mail me at JohnLeeExcelVBA@gmail.com and indicate the password "bigsky" to obtain a copy of this Microsoft Excel workbook.

Section 42.2 discusses the basic concepts of call and put options. Section 42.3 demonstrates the one period call and put option-pricing models. Section 42.4 presents the two-period option-pricing model. Section 42.5 demonstrates how to use the Microsoft Excel workbook *binomialBS_OPM.xls* to create the decision trees for a n-period Binomial Option Pricing Model. Section 42.6 demonstrates the use of the Black-Scholes model. Section 42.7 shows the relationship between the Binomial Option Pricing Model and the Black-Scholes Option Pricing Model. Finally, Sect. 42.8 shows how to use the Microsoft Excel workbook *binomialBS_OPM.xls* to demonstrate the relationship between the Binomial Option Pricing Model and the Black-Scholes Option Pricing Model.

42.2 Call and Put Options

A *call option* gives the owner the right but not the obligation to buy the underlying security at a specified price. The price in which the owner can buy the underlying price is called

J. Lee (✉)
Center for PBBEF Research, North Brunswick, NJ, USA
e-mail: johnleeexcelvba@gmail.com

C.-F. Lee et al. (eds.), *Handbook of Quantitative Finance and Risk Management*,
DOI 10.1007/978-0-387-77117-5_42, © Springer Science+Business Media, LLC 2010

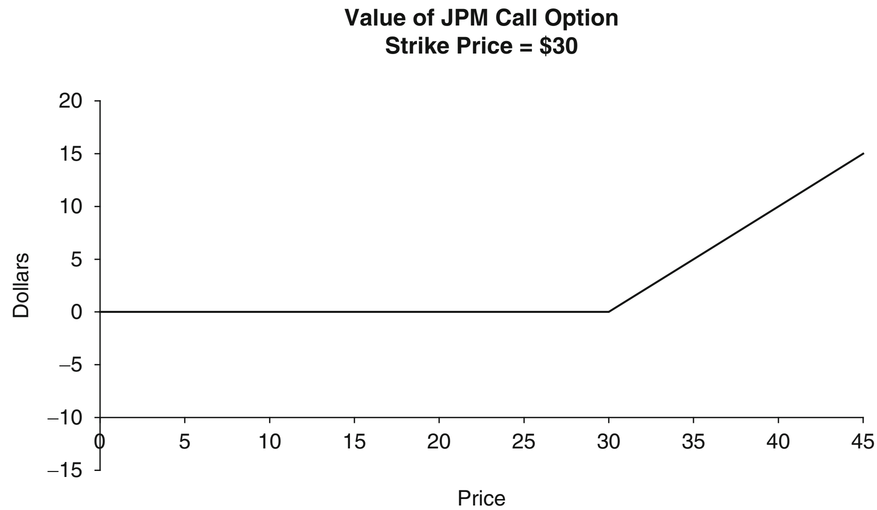


Fig. 42.1 Value of JPM Call Option

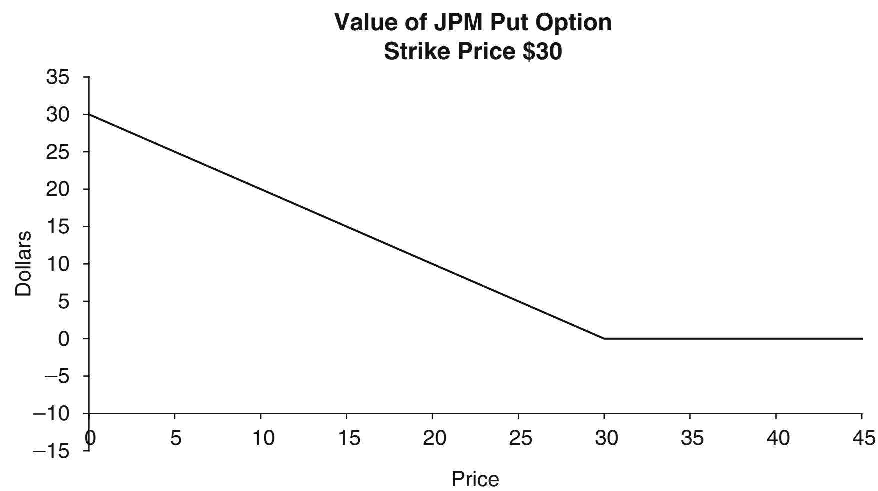


Fig. 42.2 Value of JPM Put Option

the *exercise price*. A call option becomes valuable when the exercise price is less than the current price of the underlying stock price.

For example, a call option on a JPM stock with an exercise price of $30 when the stock price of an JPM stock is $35 is worth $5. The reason it is worth $5 is because a holder of the call option can buy the JPM stock at $30 and then sell the JPM stock at the prevailing price of $35 for a profit of $5. Also, a call option on an JPM stock with an exercise price of $30 when the stock price of an JPM stock is $25 is worth zero.

A *put option* gives the owner the right but not the obligation to sell the underlying security at a specified price. A put option becomes valuable when the exercise price is more than the current price of the underlying stock price.

For example, a put option on an JPM stock with an exercise price of $30 when the stock price of a JPM stock is $25 is worth $5. The reason it is worth $5 is because a holder of the put option can buy the JPM stock at the prevailing price of $25 and then sell the JPM stock at the put price of $30 for a profit of $5. Also, a put option on a JPM stock with an exercise price of $30 when the stock price of the JPM stock is $35 is worth zero.

Figures 42.1 and 42.2 are charts showing the value of call and put options of the above JPM stock at varying prices.

42.3 One Period Option Pricing Model

What should be the value of these options? Let's look at a case where we are only concerned with the value of options for one period. In the next period a stock price can either go up or go down. Let's look at a case where we know for certain that a JPM stock with a price of $30 will either go up 5% or go down 5% in the next period and the exercise after one period is $30. Figures 42.3–42.5 shows the decision tree for the JPM stock price, the JPM call option price, and the JPM put option price, respectively.

Fig. 42.3 JPM Stock Price

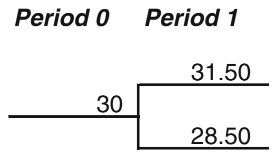


Fig. 42.4 JPM Call Option Price

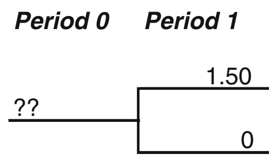
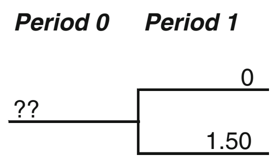


Fig. 42.5 JPM Put Option Price



Let's first consider the issue of pricing a JPM call option. Using a one-period decision tree we can illustrate the price of a JPM stock if it goes up 5% and the price of a stock JPM if it goes down 5%. Since we know the possible endings values of the JPM stock, we can derive the possible ending values of a call option. If the stock price increases to \$31.50, the price of the JPM call option will then be \$1.50 (\$31.5–\$30). If the JPM stock price declines to \$28.50, the value of the call option will worth zero because it would be below the exercise price of \$30. We have just discussed the possible ending value of a JPM call option in period 1. But, what we are really interested in is the value is of the JPM call option knowing the two resulting values of the JPM call option.

To help determine the value of a one-period JPM call option, it's useful to know that it is possible to replicate the resulting two-state of the value of the JPM call option by buying a combination of stocks and bonds. The formula below replicates the situation where the price increases to \$31.50. We will assume that the interest rate for the bond is 3%.

$$31.5\text{S} + 1.03\text{B} = 1.5$$
$$28.5\text{S} + 1.03\text{B} = 0$$

We can use simple algebra to solve for both S and B. The first thing that we need to do is to rearrange the second equation as follows,

$$1.07\text{B} = -28.5\text{S}$$

With the above equation, we can rewrite the first equation as

$$31.5\text{S} + (-28.5\text{S}) = 1.5$$
$$3\text{S} = 1.5$$
$$\text{S} = 0.5$$

We can solve for B by substituting the value 0.5 for S in the first equation.

$$31.5\,(0.5) + 1.03\text{B} = 1.5$$
$$15.75 + 1.03\text{B} = 1.5$$
$$1.03\text{B} = -14.25$$
$$\text{B} = -13.8350$$

Therefore, from the above simple algebraic exercise, we should at period 0 buy 0.5 shares of JPM stock and borrow 13.8350 at 3% to replicate the payoff of the JPM call option. This means the value of a JPM call option should be $0.5 \times 30 - 13.8350 = 1.165$.

If this were not the case, there would then be arbitrage profits. For example, if the call option were sold for \$3 there would be a profit of 1.835. This would result in an increase in the selling of the JPM call option. The increase in the supply of JPM call options would push the price down for the call options. If the call option were sold for \$1, there would be a saving of 0.165. This saving would result in the increase demand for the JPM call option. This increase demand would result in an increase of the price of the call option. The equilibrium point would be 1.165.

Using the above mentioned concept and procedure, Benninga (2000) has derived a one-period call option model as

$$C = q_u \text{Max}[S(1+u)X, 0] + q_d \text{Max}[S(1+d) - X, 0] \tag{42.1}$$

where,

$$q_u = \frac{i-d}{(1+i)(u-d)}$$
$$q_d = \frac{u-i}{(1+i)(u-d)}$$
$$u = \text{increase factor}$$
$$d = \text{down factor}$$
$$i = \text{interest rate}$$

If we let $i = r$, $p = (r-d)/(u-d)$, $1-p = (u-r)/(u-d)$, $R = 1/(1+r)$, $C_u = \text{Max}[S(1+u) - X, 0]$ and $C_d = \text{Max}[S(1+d) - X, 0]$ then we have

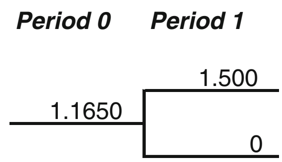
$$C = [pC_u + (1-p)C_d]/R, \tag{42.2}$$

where, Cu = call option price after increase
Cd = call option price after decrease

Equation (42.2) is identical to Equation (6B.6) in Lee et al. (2000, p. 234).[1]

[1] Please note that in Lee et al. (2000, p. 234) $u = 1+$ percentage of price increase, $d = 1-$ percentage of price increase.

Fig. 42.6 Call Option Price



See below to calculate the value of the above one-period call option where the strike price, X, is \$30 and the risk free interest rate is 3%. We will assume that the price of a stock for any given period will either increase or decrease by 5%.

$$X = \$30$$
$$S = \$30$$
$$u = 1.05$$
$$d = 0.95$$
$$R = 1 + r = 1 + 0.03$$
$$p = (1.03 - 0.95)/(1.05 - 0.95)$$
$$C = [0.8(1.5) + 0.2(0)]/1.03 = \$1.1650$$

Therefore from the above calculations, the value of the call option is \$7.94. Figure 42.6 shows the resulting decision tree for the above call option.

Like the call option, it is possible to replicate the resulting two states of the value of the put option by buying a combination of stocks and bonds. See below is for the formula to replicate the situation where the price decreases to \$28.50.

$$31.5\text{S} + 1.03\text{B} = 0$$
$$28.5\text{S} + 1.03\text{B} = 1.5$$

We will use simple algebra to solve for both S and B. The first thing we will do is to rewrite the second equation as follows,

$$1.03\text{B} = 1.5 - 28.5\text{S}$$

The next thing is to substitute the above equation to the first put option equation. Doing this would result in the following,

$$31.5\text{S} + 1.5 - 28.5\text{S} = 0$$

The following solves for S,

$$3\text{S} = -1.5$$
$$\text{S} = -0.5$$

Now let us solve for B by putting the value of S into the first equation. This is shown below.

$$31.5\,(-0.5) + 1.03\text{B} = 0$$
$$1.03\text{B} = 15.75$$
$$\text{B} = 15.2913$$

From the above simple algebra exercise we have S $= -0.5$ and B $= 15.2913$. This tells us that we should in period 0 lend \$15.2913 at 3% and sell 0.5 shares of stock to replicate the put option payoff for period 1. And the value of the JPM put option should be $30(-0.5) + 15.2913 = 0.2913$.

Using the same arbitrage argument that we used in the discussing the call option, 0.2913 has to be the equilibrium price of the put option.

As with the call option, Benninga (2000) has derived a one-period put option model as

$$P = q_u \text{Max}[X - S(1 + u), 0] + q_d \text{Max}[X - S(1 + d), 0] \tag{42.3}$$

where,

$$q_u = \frac{i - d}{(1 + i)(u - d)}$$
$$q_d = \frac{u - i}{(1 + i)(u - d)}$$
$$u = \text{increase factor}$$
$$d = \text{down factor}$$
$$i = \text{interest rate}$$

If we let $i = r$, $p = (r-d)/(u-d)$, $1-p = (u-r)/(u-d)$, $R = 1/(1 + r)$, $P_u = \text{Max}[X - S(1 + u), 0]$ and $P_d = \text{Max}[X - S(1 + d), 0]$ then we have

$$P = [pP_u + (1 - p)P_d]/R, \tag{42.4}$$

where, P_u = put option price after increase

P_d = put option price after decrease

Below calculates the value of the above oneperiod put option where the strike price, X, is \$30 and the risk-free interest rate is 3%.

$$P = [0.8\,(0) + 0.2\,(1.03)]/1.03 = \$.2913$$

From the above calculation the put option pricing decision tree would look like the following.

Figure 42.7 shows the resulting decision tree for the above put option.

There is a relationship between the price of a put option and the price of a call option. This relationship is called the put-call parity. Equation (42.5) shows the relationship between the price of a put option and the price of a call option.

$$\text{P} = \text{C} + \text{X/R} - \text{S} \tag{42.5}$$

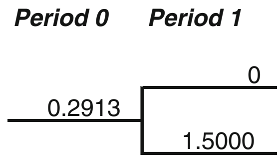


Fig. 42.7 JPM Put Option Price

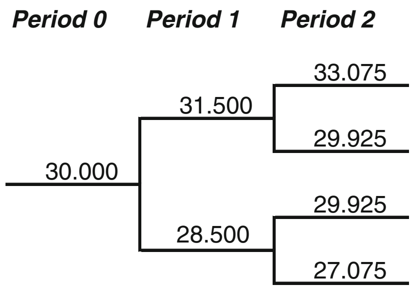


Fig. 42.8 JPM Stock Price

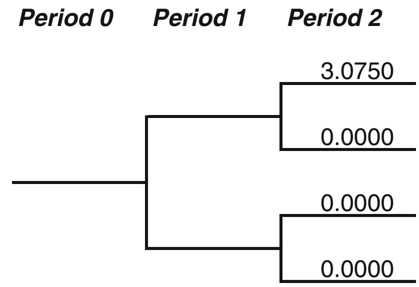


Fig. 42.9 JPM Call Option

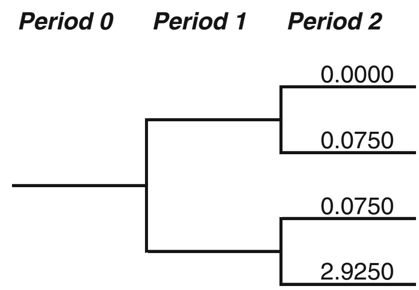


Fig. 42.10 JPM Put Option

Where,

C = call price
X = strike price
R = 1+ interest rate
S = Stock Price

The following uses the put-call parity to calculate the price of the JPM put option.

$$\begin{aligned} P &= \$1.165 + \$30/\,(1.03)\; - \$30 \\ &= 1.165 + 9.1262 - 30 \\ &= .2912 \end{aligned}$$

42.4 Two-Period Option Pricing Model

We now will look at pricing options for two periods. Figure 42.8 shows the stock price decision tree based on the parameters indicated in the last section. This decision tree was created based on the assumption that a stock price will either increase or decrease by 10%.

How do we price the value of a call and put option for two periods?

The highest possible value for our stock based on our assumption is \$33.075. We get this value first by multiplying the stock price at period 0 by 105% to get the resulting value of \$31.50 of period 1. We then multiply the stock price in period 1 by 105% to get the resulting value of \$33.075. In period two, the value of a call option when a stock price is \$33.075 is the stock price minus the exercise price, \$33.075 − 30, or \$3.075. In period two, the value of a put option when a stock price of \$33.075 is the exercise price minus the stock price, \$30 – \$33.075, or −\$3.075. A negative value has no value to an investor so the value of the put option would be zero.

The lowest possible value for our stock based on our assumptions is \$27.075. We get this value first by multiplying the stock price at period 0 by 95% (decreasing the value of the stock by 5%) to get the resulting value of \$28.5 of period 1. We then again multiply the stock price in period 1 by 95% to get the resulting value of \$27.075. In period two, the value of a call option when a stock price is \$27.075 is the stock price minus the exercise price, \$27.075 − \$30, or −\$2.925. A negative value has no value to an investor so the value of a call option would be zero. In period two, the value of a put option when a stock price is \$27.075 is the exercise price minus the stock price, \$30 – \$27.075, or \$2.925. We can derive the call and put option value for the other possible value of the stock in period 2 in the same fashion.

Figures 42.9 and 42.10 show the possible call and put option values for period 2.

We cannot calculate the value of the call and put option in period 1 the same way we did in period 2 because it's not the ending value of the stock. In period 1 there are two possible call values. One value is when the stock price increased and one value is when the stock price decreased. The call option decision tree shown in Fig. 42.9 shows two possible values for a call option in period 1. If we just focus on the value of a call option when the stock price increases from period one, we will notice that it is like the decision tree for a call option for one period. This is shown in Fig. 42.11.

Using the same method for pricing a call option for one period, the price of a call option when stock price increase

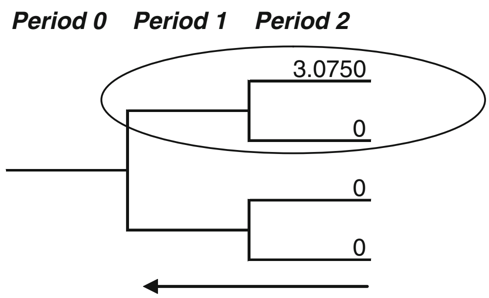


Fig. 42.11 JPM Call Option

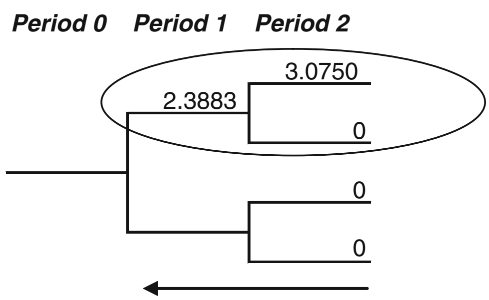


Fig. 42.12 JPM Call Option

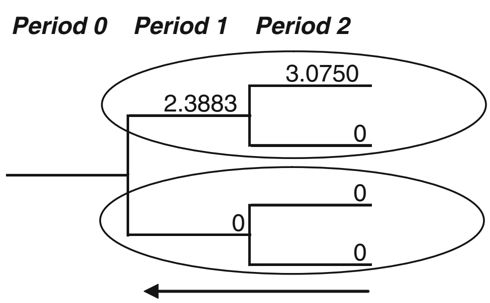


Fig. 42.13 JPM Call Option

from period 0 will be \$2.3883. The resulting decision tree is shown in Fig. 42.12.

In the same fashion we can price the value of a call option when a stock price decreases. The price of a call option when a stock price decreases from period 0 is zero. The resulting decision tree is shown in Fig. 42.13.

In the same fashion we can price the value of a call option in period 0. The resulting decision tree is shown in Fig. 42.14.

We can calculate the value of a put option in the same manner as we did in calculating the value of a call option. The decision tree for a put option is shown in Fig. 42.15.

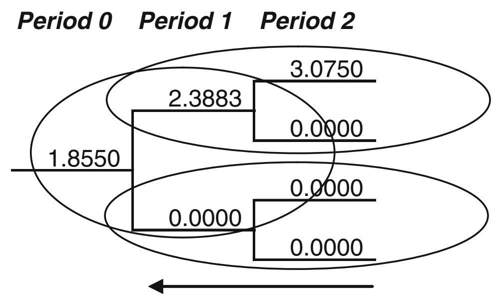


Fig. 42.14 JPM Call Option

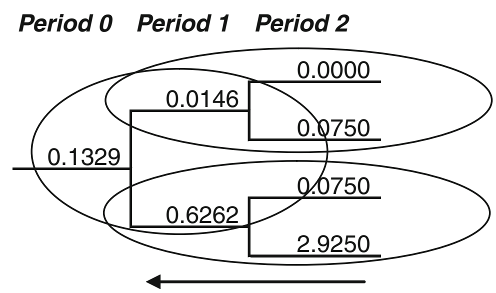


Fig. 42.15 JPM Put Option

Table 42.1 Binomial calculations

Periods	Calculations
1	3
2	7
3	17
4	31
5	63
6	127
7	255
8	511
9	1, 023
10	2, 047
11	4, 065
12	8, 191

42.5 Using Microsoft Excel to Create the Binomial Option Trees

In the previous section we priced the value of a call and put option by pricing backwards, from the last period to the first period. This method of pricing call and put options will work for any n-period. To price the value of a call options for two periods required seven sets of calculations. The number of calculations increases dramatically as n increases. Table 42.1 lists the number of calculations for specific number of periods.

After two periods it becomes very cumbersome to calculate and create the decision trees for call and put options. In the previous section we saw that the calculations were very repetitive and mechanical. To solve this problem, we will use

Microsoft Excel to do the calculations and create the decision trees for the call and put options. We will also use Microsoft Excel to calculate and draw the related Decision Trees for the underlying stocks and bonds.

To avoid the repetitive and mechanical calculation of the Binomial Option Pricing Model, we will look at a Microsoft Excel file called *binomialBS_OPM.xls*. And use the workbook to produce four decision trees for the JPM stock which was discussed in the previous sections. The four decision trees are as follows:

1. Stock Price
2. Call Option Price
3. Put Option Price
4. Bond Price

Figure 42.16 shows the Excel file *binomialBS_OPM.xls* after the file is opened. Pushing the button shown in Fig. 42.16 will get the dialog box shown in Fig. 42.17.

The dialog box shown in Fig. 42.17 shows the parameters for the *Binomial Option Pricing Model*. These parameters are changeable. The dialog box in Fig. 42.17 shows the default values.

Pushing the *calculate* button shown in Fig. 42.17 will produce the four decision trees shown in Figs. 42.18–42.21.

The table at the beginning of this section indicated 31 calculations were required to create a decision tree that has four periods. This section showed four decision trees. Therefore, the Excel file did $31 \times 4 = 124$ calculations to create the four decision trees.

Benninga (2000, p. 260) has defined the price of a call option in a Binomial Option Pricing Model with n periods as

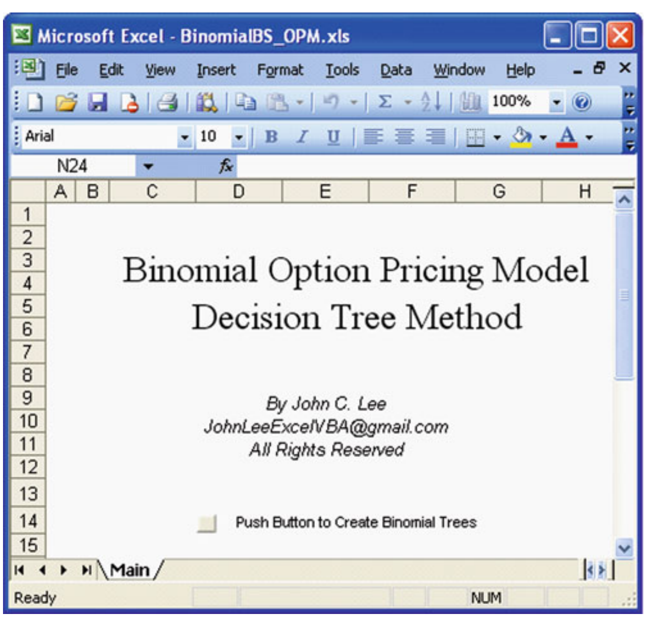


Fig. 42.16 Excel File BinomialBS_OPM.xls

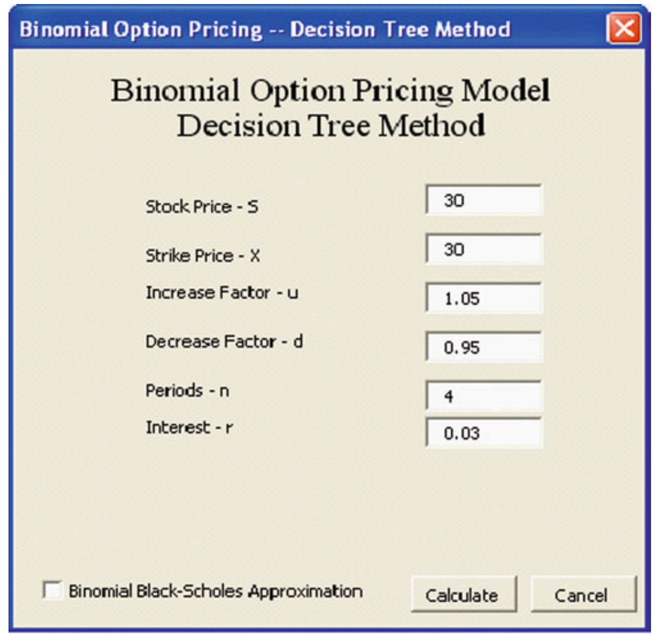


Fig. 42.17 Dialog Box Showing Parameters for the Binomial Option Pricing Model

$$C = \sum_{i=0}^{n} \binom{n}{i} q_u^i q_d^{n-i} \max[S(1+u)^i (1+d)^{n-i}, 0] \qquad (42.6)$$

and the price of a put option in a Binomial Option Pricing Model with n-periods as

$$P = \sum_{i=0}^{n} \binom{n}{i} q_u^i q_d^{n-i} \max[X - S(1+u)^i (1+d)^{n-i}, 0] \qquad (42.7)$$

Lee et al. (2000, p. 237) has defined the pricing of a call option in a Binomial

Option Pricing Model with n-period as,

$$C = \frac{1}{R^n} \sum_{k=0}^{n} \frac{n!}{k!(n-k!)} p^k (1-p)^{n-k} \times \max\left[0, (1+u)^k (1+d)^{n-k}, S - X\right] \qquad (42.8)$$

The definition of the pricing of a put option in a Binomial Option Pricing Model

with n period would then be defined as,

$$P = \frac{1}{R^n} \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \times \max\left[0, X - (1+u)^k (1+d)^{n-k}, S\right] \qquad (42.9)$$

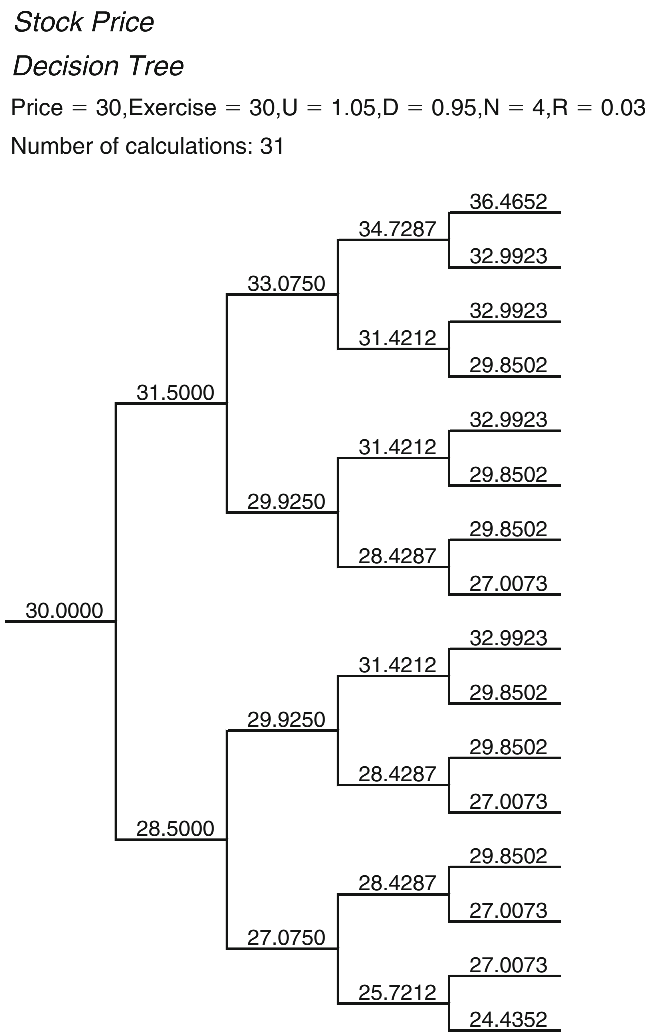


Fig. 42.18 Stock Price Decision Tree

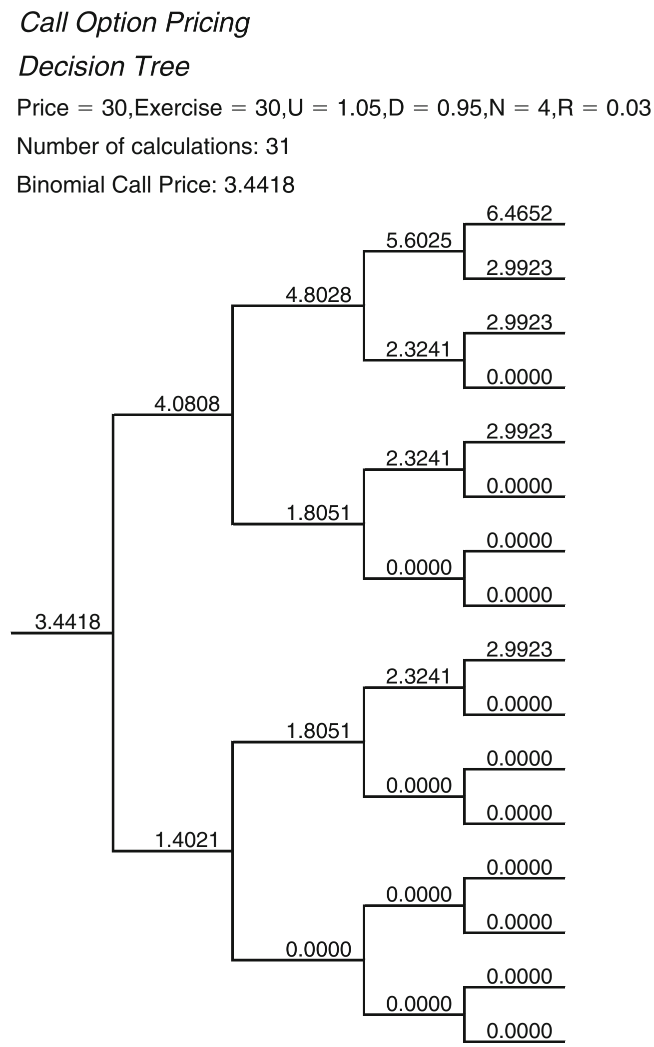


Fig. 42.19 Call Option Pricing Decision Tree

42.6 Black-Scholes Option Pricing Model

The most famous option pricing model is the Black-Scholes Option Pricing Model. In this section we will demonstrate the usage of the Black-Scholes Option Pricing Model. In later sections we will demonstrate the relationship between the Binomial Option Pricing Model and the Black-Scholes Pricing Model. The Black-Scholes Model prices European call and put options. The Black-Scholes model for a European call option is

$$C = SN(d1) - Xe^{-rT}N(d2) \tag{42.10}$$

Where,

C = Call price
S = Stock price
r = risk free interest rate
T = time to maturity of option in years
N() = standard normal distribution
σ = stock volatility

$$d1 = \frac{\ln(S/X) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

$$d2 = d1 - \sigma\sqrt{T}$$

Let's manually calculate the price of an European call option in terms of Equation (42.10) with the following parameter values, S = 30, X = 30, r = 3%, T = 4, σ = 20%:

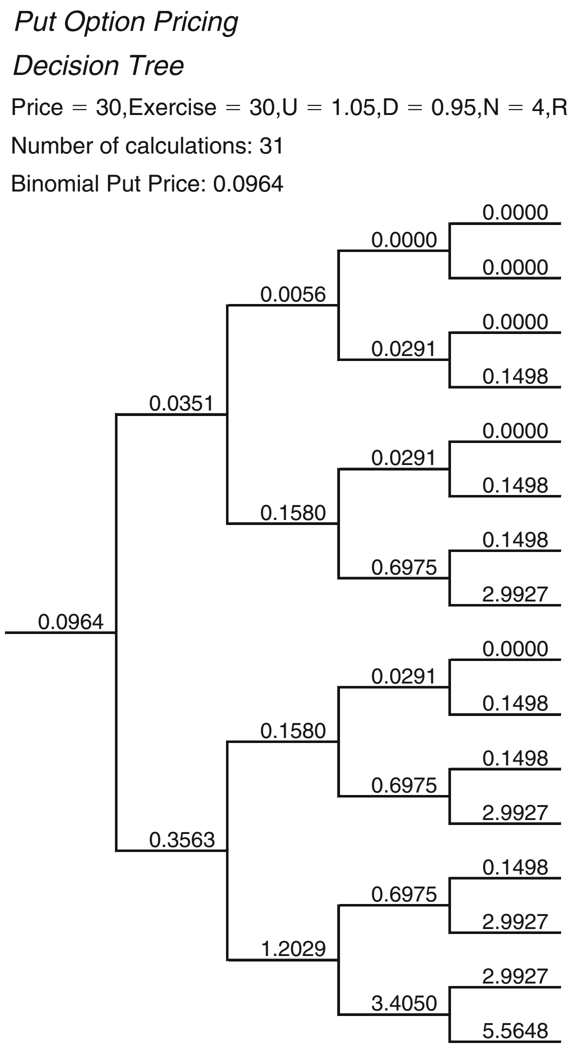


Fig. 42.20 Put Option Pricing Decision Tree

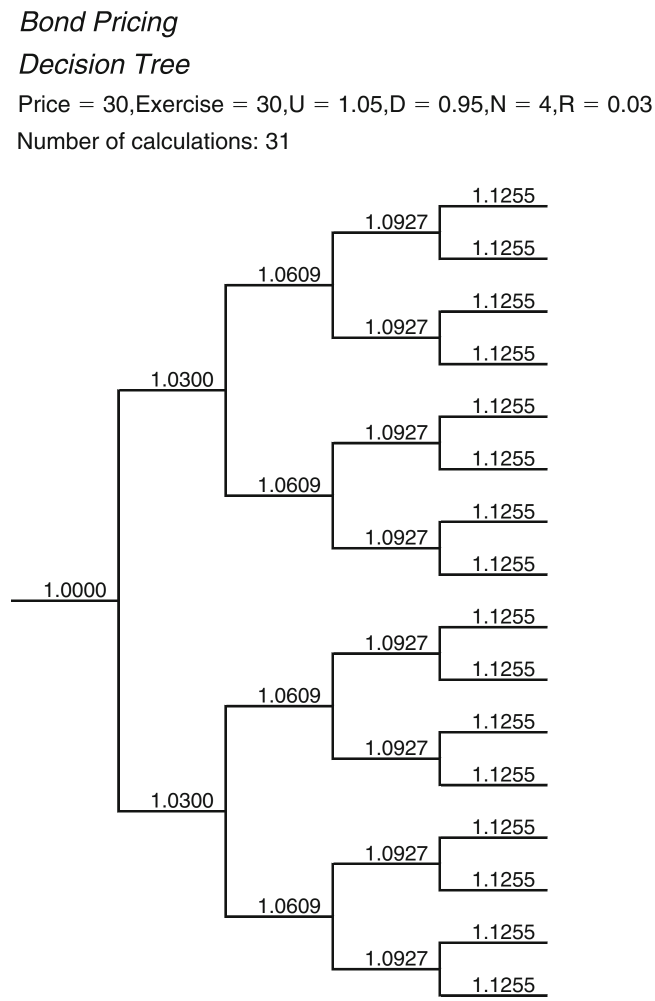


Fig. 42.21 Bond Pricing Decision Tree

Solution,

$$
\begin{aligned}
d1 &= \frac{\ln(S/X) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \\
&= \frac{\ln(30/30) + \left(.03 + .\frac{2^2}{2}\right)(4)}{.2\sqrt{4}} \\
&= \frac{(.03 + .02) * 4}{.4} = \frac{.2}{.4} = .5, \\
d2 &= .5 - .2\sqrt{4} = .1
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{N}\,(\mathrm{d}1) = 0.69146, \mathrm{N}\,(\mathrm{d}2) = 0.5398, e^{-rT} = 0.8869 \\
\mathrm{C} = (30) * (0.69146) - (30) * (0.8869) * 0.5398 \\
= 20.7438 - 14.3624 = 6.3813414
\end{aligned}
$$

The Black-Scholes put-call parity equation is

$$P = C - S + Xe^{-rT}$$

The put option value for the stock would be

$$P = 6.38 - 30 + 30\,(0.8869) = 2.987$$

42.7 Relationship Between the Binomial OPM and the Black-Scholes OPM

We can use either the Binomial model or Black-Scholes to price an option. They both should result in similar numbers. If we look at the parameters in both models we will notice that the Binomial model has an *Increase Factor (U)*, a *Decrease Factor (D)* and n-period parameters that the Black-Scholes model does not have. We also notice that the Black-Scholes model has the σ and T parameters whereas the Binomial model does not Benninga (2008) suggests the following translation between the Binomial and Black-Scholes parameters.

$$\Delta t = T/n \quad R = e^{r\Delta t} \quad U = e^{\sigma\sqrt{\Delta t}} \quad D = e^{-\sigma\sqrt{\Delta t}}$$

In the Excel program shown in Appendix 42A, we use Benniga's (2008) *Increase Factor* and *Decrease Factor* definitions. They are defined as follows:

$$q_U = \frac{R - D}{R(U - D)}, \quad q_D = \frac{U - R}{R(U - D)}$$

where,

U = 1+ percentage of price increase
D = 1− percentage of price increase
R = 1+ interest rate

42.8 Decision Tree Black-Scholes Calculation

We will now use the *BinomialBS_OPM.xls* Excel file to calculate the Binomial and Black-Scholes call and put values illustrated in Sect. 42.5. Note that in Fig. 42.22. the *Binomial*

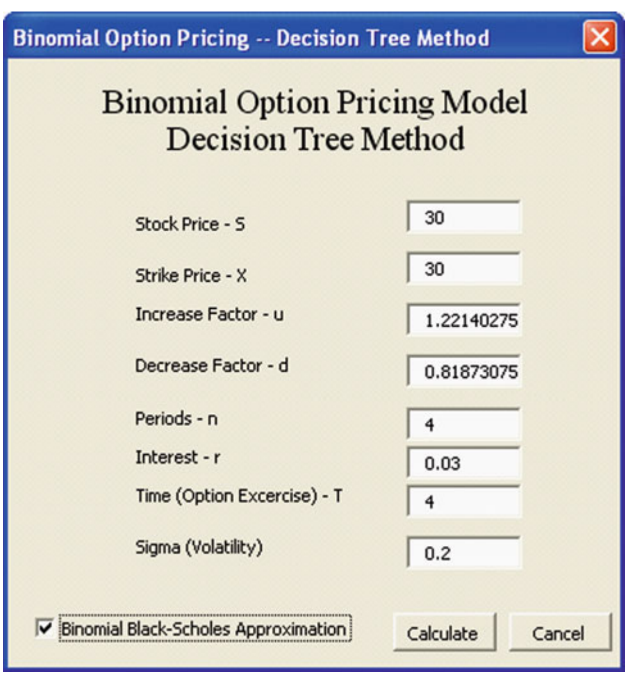


Fig. 42.22 Dialog Box Showing Parameters for the Binomial Option Pricing Model

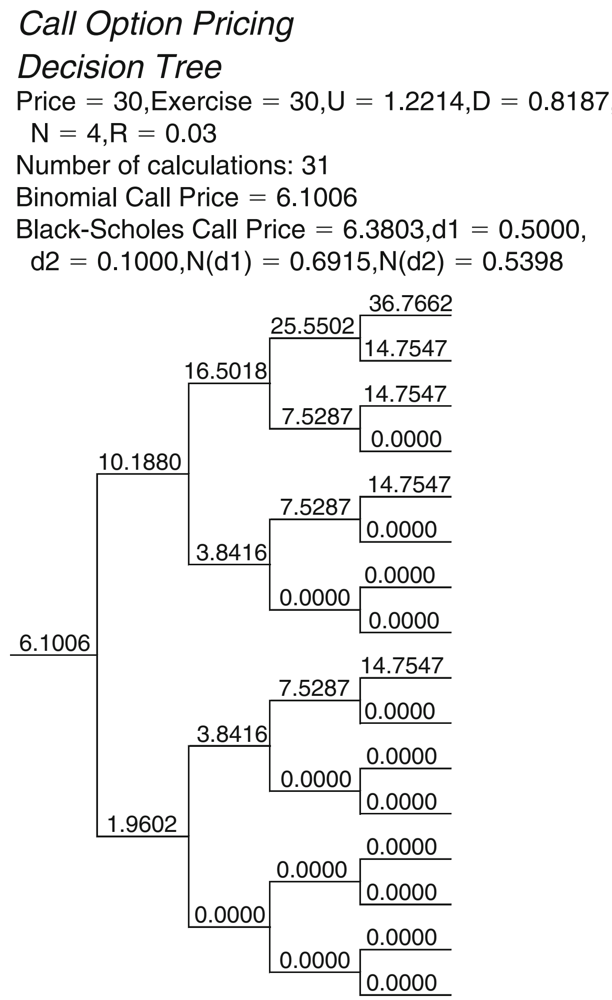


Fig. 42.23 Decision Tree approximation of Black-Scholes Call Pricing

Black-Scholes Approximation box is checked. Checking this box will cause T and Sigma parameters to appear and will adjust the *Increase Factuor – u and Decrease Factor – d* parameters. The adjustment was done as indicated in Sect. 42.7.

Note in Figs. 42.23 and 42.24 the Binomial Option Pricing Model value does not agree with the Black-Scholes Option Pricing model. The Binomial OPM value will get very close to the Black-Scholes OPM value once the Binomial parameter n becomes very large. Benninga (2008) demonstrated that the Binomial value will be close to the Black-Scholes when the Binomial n parameter is larger than 500.

42.9 Conclusion

This paper has demonstrated, using Microsoft Excel and decision trees, the Binomial Option Model in a less mathematical fashion. This paper allowed the reader to focus more

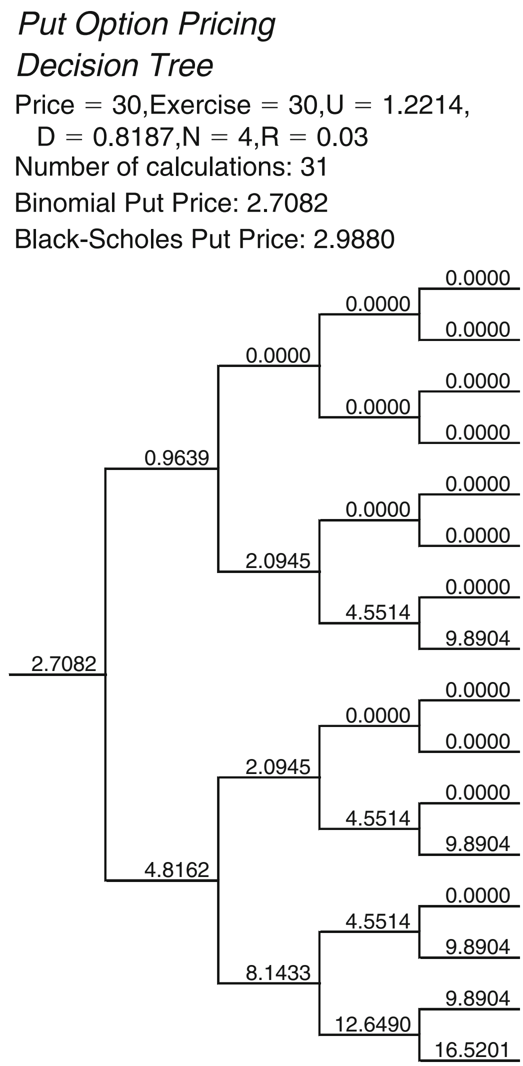


Fig. 42.24 Decision Tree approximation of Black-Scholes Put Pricing

which were created by Microsoft Excel. This paper also demonstrates that using Microsoft Excel releases the reader from the computation burden of the Binomial Option Model and allows a better focus on the concepts by studying the associated decision trees.

This paper also published the Microsoft Excel Visual Basic for Application (VBA) code that created the Binomial Option Decision Trees. In addition, a major computer science programming concept used by the Excel VBA program in this paper is recursive programming. Recursive programming follows a procedure which repeats itself many times. Inside the procedure are statements to decide when not to call itself.

This paper also used decision trees to demonstrate the relationship between the Binomial Option Pricing Model and the Black-Scholes Option Pricing Model.

References

Benninga, S. 2000. *Financial modeling*. MIT Press, Cambridge.
Benninga, S. 2008. *Financial modeling*. MIT Press, Cambridge.
Black, F. and M. Scholes. 1973. "The pricing of options and corporate liabilities." *Journal of Political Economy* 31, 637–659.
Cox, J., S. A. Ross, and M. Rubinstein. 1979. "Option pricing: a simplified approach." *Journal of Financial Economics* 7, 229–263.
Lee, C. F., J. C. Lee, and A. C. Lee. 2000. *Statistics for business and financial economics*. World Scientific, NJ.
Rendleman, R. J., Jr., and B. J. Barter. 1979. "Two-state option pricing." *Journal of Finance*, 34(5), 1093–1110.
Wells, E. and S.Harshbarger, 1997. *Microsoft Excel 97 Developer's Handbook*. Microsoft Press, Redmond.

Appendix 42A Excel VBA Code–Binomial Option Pricing Model

It is important to note that the thing that makes Microsoft Excel powerful is that it offers a powerful professional programming language called Visual Basic for Applications (VBA). This section shows the VBA code that generated the Decision Trees for the Binomial Option pricing model. This code is in the form *frmBinomiaOption*. The procedure *cmdCalculate_Click* is the first procedure to run.

Decision Tree Visual Basic for Application Programming Code

```
'/***************************************************************************
'/         Relationship Between the Binomial OPM
'/            and Black-Scholes OPM:
'/            Decision Tree and Microsoft Excel Approach
'/
'/                     by John Lee
'/                     JohnLeeExcelVBA@gmail.com
'/                     All Rights Reserved
'/***************************************************************************
Option Explicit
Dim mwbTreeWorkbook As Workbook
Dim mwsTreeWorksheet As Worksheet
Dim mwsCallTree As Worksheet
Dim mwsPutTree As Worksheet
Dim mwsBondTree As Worksheet
Dim mdblPFactor As Double
Dim mBinomialCalc As Long
Dim mCallPrice As Double   'jcl 12/8/2008
Dim mPutPrice As Double   'jcl 12/8/2008

'/**************************************************
'/Purpose:  Keep track the numbers of binomial calc
'/*************************************************
Property Let BinomialCalc(l As Long)
    mBinomialCalc = l
End Property
Property Get BinomialCalc() As Long
    BinomialCalc = mBinomialCalc
End Property
Property Set TreeWorkbook(wb As Workbook)
    Set mwbTreeWorkbook = wb
End Property
Property Get TreeWorkbook() As Workbook
    Set TreeWorkbook = mwbTreeWorkbook
End Property
Property Set TreeWorksheet(ws As Worksheet)
    Set mwsTreeWorksheet = ws
End Property
Property Get TreeWorksheet() As Worksheet
    Set TreeWorksheet = mwsTreeWorksheet
End Property
Property Set CallTree(ws As Worksheet)
    Set mwsCallTree = ws
End Property
Property Get CallTree() As Worksheet
    Set CallTree = mwsCallTree
End Property
Property Set PutTree(ws As Worksheet)
    Set mwsPutTree = ws
End Property
Property Get PutTree() As Worksheet
    Set PutTree = mwsPutTree
End Property
Property Set BondTree(ws As Worksheet)
    Set mwsBondTree = ws
End Property
Property Get BondTree() As Worksheet
    Set BondTree = mwsBondTree
End Property
Property Let CallPrice(dCallPrice As Double)
    '12/8/2008
    mCallPrice = dCallPrice
End Property
```

(continued)

```
Property Get CallPrice() As Double
    Let CallPrice = mCallPrice
End Property
Property Let PutPrice(dPutPrice As Double)
    '12/10/2008
    mPutPrice = dPutPrice
End Property
Property Get PutPrice() As Double
    '12/10/2008
    Let PutPrice = mPutPrice
End Property
Property Let PFactor(r As Double)
    Dim dRate As Double

    dRate = ((1 + r) - Me.txtBinomialD) / (Me.txtBinomialU - Me.txtBinomialD)

    Let mdblPFactor = dRate
End Property
Property Get PFactor() As Double
    Let PFactor = mdblPFactor
End Property

Property Get qU() As Double
    Dim dblDeltaT As Double
    Dim dblDown As Double
    Dim dblUp As Double
    Dim dblR As Double

    dblDeltaT = Me.txtTimeT / Me.txtBinomialN
    dblR = Exp(Me.txtBinomialr * dblDeltaT)
    dblUp = Exp(Me.txtSigma * VBA.Sqr(dblDeltaT))
    dblDown = Exp(-Me.txtSigma * VBA.Sqr(dblDeltaT))


    qU = (dblR - dblDown) / (dblR * (dblUp - dblDown))
End Property

Property Get qD() As Double

    Dim dblDeltaT As Double
    Dim dblDown As Double
    Dim dblUp As Double
    Dim dblR As Double

    dblDeltaT = Me.txtTimeT / Me.txtBinomialN
    dblR = Exp(Me.txtBinomialr * dblDeltaT)
    dblUp = Exp(Me.txtSigma * VBA.Sqr(dblDeltaT))
    dblDown = Exp(-Me.txtSigma * VBA.Sqr(dblDeltaT))

    qD = (dblUp - dblR) / (dblR * (dblUp - dblDown))
End Property


Private Sub chkBinomialBSApproximation_Click()
    On Error Resume Next
        'Time and Sigma only BlackScholes parameter
        Me.txtTimeT.Visible = Me.chkBinomialBSApproximation
        Me.lblTimeT.Visible = Me.chkBinomialBSApproximation
        Me.txtSigma.Visible = Me.chkBinomialBSApproximation
        Me.lblSigma.Visible = Me.chkBinomialBSApproximation
        txtTimeT_Change

End Sub
```

(continued)

```
Private Sub cmdCalculate_Click()
    Me.Hide
    BinomialOption
    Unload Me
End Sub

Private Sub cmdCancel_Click()
    Unload Me
End Sub


Private Sub txtBinomialN_Change()
    'jcl 12/8/2008
    On Error Resume Next
    If Me.chkBinomialBSApproximation Then
        Me.txtBinomialU = Exp(Me.txtSigma * Sqr(Me.txtTimeT / Me.txtBinomialN))
        Me.txtBinomialD = Exp(-Me.txtSigma * Sqr(Me.txtTimeT / Me.txtBinomialN))
    End If
End Sub


Private Sub txtTimeT_Change()

    'jcl 12/8/2008
    On Error Resume Next
    If Me.chkBinomialBSApproximation Then
        Me.txtBinomialU = Exp(Me.txtSigma * Sqr(Me.txtTimeT / Me.txtBinomialN))
        Me.txtBinomialD = Exp(-Me.txtSigma * Sqr(Me.txtTimeT / Me.txtBinomialN))
    End If
End Sub

Private Sub UserForm_Initialize()

   With Me

        .txtBinomialS = 30
        .txtBinomialX = 30
        .txtBinomialD = 0.95
        .txtBinomialU = 1.05
        .txtBinomialN = 4
        .txtBinomialr = 0.03
        .txtSigma = 0.2
        .txtTimeT = 4

        Me.chkBinomialBSApproximation = False
    End With

    chkBinomialBSApproximation_Click
    Me.Hide
End Sub

Sub BinomialOption()
    Dim wbTree As Workbook
    Dim wsTree As Worksheet
    Dim rColumn As Range
    Dim ws As Worksheet

    Set Me.TreeWorkbook = Workbooks.Add
    Set Me.BondTree = Me.TreeWorkbook.Worksheets.Add
    Set Me.PutTree = Me.TreeWorkbook.Worksheets.Add
    Set Me.CallTree = Me.TreeWorkbook.Worksheets.Add
    Set Me.TreeWorksheet = Me.TreeWorkbook.Worksheets.Add
```

(continued)

```
    Set rColumn = Me.TreeWorksheet.Range("a1")

    With Me
        .BinomialCalc = 0
        .PFactor = Me.txtBinomialr
        .CallTree.Name = "Call Option Price"
        .PutTree.Name = "Put Option Price"
        .TreeWorksheet.Name = "Stock Price"
        .BondTree.Name = "Bond"
    End With

    DecisionTree rCell:=rColumn, nPeriod:=Me.txtBinomialN + 1, _
                dblPrice:=Me.txtBinomialS, sngU:=Me.txtBinomialU, _
                sngD:=Me.txtBinomialD

    DecitionTreeFormat
    TreeTitle wsTree:=Me.TreeWorksheet, sTitle:="Stock Price "
    TreeTitle wsTree:=Me.CallTree, sTitle:="Call Option Pricing"
    TreeTitle wsTree:=Me.PutTree, sTitle:="Put Option Pricing"
    TreeTitle wsTree:=Me.BondTree, sTitle:="Bond Pricing"

    Application.DisplayAlerts = False
    For Each ws In Me.TreeWorkbook.Worksheets
        If Left(ws.Name, 5) = "Sheet" Then
            ws.Delete
        Else
            ws.Activate
            ActiveWindow.DisplayGridlines = False
            ws.UsedRange.NumberFormat = "#,##0.0000_);(#,##0.0000)"
        End If
    Next
    Application.DisplayAlerts = True
    Me.TreeWorksheet.Activate
End Sub
Sub TreeTitle(wsTree As Worksheet, sTitle As String)
    wsTree.Range("A1:A5").EntireRow.Insert (xlShiftDown)
    With wsTree
        With .Cells(1)
               .Value = sTitle
               .Font.Size = 20
               .Font.Italic = True
        End With
        With .Cells(2, 1)
             .Value = "Decision Tree"
             .Font.Size = 16
             .Font.Italic = True
        End With
        With .Cells(3, 1)
             .Value = "Price = " & Me.txtBinomialS & _
                    ",Exercise = " & Me.txtBinomialX & _
                    ",U = " & Format(Me.txtBinomialU, "#,##0.0000") & _
                    ",D = " & Format(Me.txtBinomialD, "#,##0.0000") & _
                    ",N = " & Me.txtBinomialN & _
                    ",R = " & Me.txtBinomialr
             .Font.Size = 14
        End With
        With .Cells(4, 1)
            .Value = "Number of calculations: " & Me.BinomialCalc
            .Font.Size = 14
        End With
```

(continued)

```
        If wsTree Is Me.CallTree Then
            With .Cells(5, 1)
                .Value = "Binomial Call Price= " & Format(Me.CallPrice, "#,##0.0000")
                .Font.Size = 14
            End With

          If Me.chkBinomialBSApproximation Then
            wsTree.Range("A6:A7").EntireRow.Insert (xlShiftDown)

            With .Cells(6, 1)
                .Value = "Black-Scholes Call Price= " & Format(Me.BS_Call, "#,##0.0000") _

                          & ",d1=" & Format(Me.BS_D1, "#,##0.0000") _
                          & ",d2=" & Format(Me.BS_D2, "#,##0.0000") _
                          & ",N(d1)=" & Format(WorksheetFunction.NormSDist(BS_D1), "#,##0.0000") _
                          & ",N(d2)=" & Format(WorksheetFunction.NormSDist(BS_D2), "#,##0.0000")
                .Font.Size = 14
            End With
           End If
        ElseIf wsTree Is Me.PutTree Then
            With .Cells(5, 1)
                .Value = "Binomial Put Price: " & Format(Me.PutPrice, "#,##0.0000")
                .Font.Size = 14
            End With
           If Me.chkBinomialBSApproximation Then
            wsTree.Range("A6:A7").EntireRow.Insert (xlShiftDown)

            With .Cells(6, 1)
                .Value = "Black-Scholes Put Price: " & Format(Me.BS_PUT, "#,##0.0000")
                .Font.Size = 14
            End With
           End If
        End If

    End With
End Sub
Sub BondDecisionTree(rPrice As Range, arCell As Variant, iCount As Long)
    Dim rBond As Range
    Dim rPup As Range
    Dim rPDown As Range

    Set rBond = Me.BondTree.Cells(rPrice.Row, rPrice.Column)
    Set rPup = Me.BondTree.Cells(arCell(iCount - 1).Row, arCell(iCount - 1).Column)
    Set rPDown = Me.BondTree.Cells(arCell(iCount).Row, arCell(iCount).Column)

    If rPup.Column = Me.TreeWorksheet.UsedRange.Columns.Count Then
        rPup.Value = (1 + Me.txtBinomialr) ^ (rPup.Column - 1)
        rPDown.Value = rPup.Value
    End If

    With rBond
        .Value = (1 + Me.txtBinomialr) ^ (rBond.Column - 1)
        .Borders(xlBottom).LineStyle = xlContinuous
    End With
            rPDown.Borders(xlBottom).LineStyle = xlContinuous
            With rPup
                .Borders(xlBottom).LineStyle = xlContinuous
                .Offset(1, 0).Resize((rPDown.Row - rPup.Row), 1). _
                    Borders(xlEdgeLeft).LineStyle = xlContinuous
            End With

End Sub
```

(continued)

```
Sub PutDecisionTree(rPrice As Range, arCell As Variant, iCount As Long)
    Dim rCall As Range
    Dim rPup As Range
    Dim rPDown As Range

    Set rCall = Me.PutTree.Cells(rPrice.Row, rPrice.Column)
    Set rPup = Me.PutTree.Cells(arCell(iCount - 1).Row, arCell(iCount - 1).Column)
    Set rPDown = Me.PutTree.Cells(arCell(iCount).Row, arCell(iCount).Column)

    If rPup.Column = Me.TreeWorksheet.UsedRange.Columns.Count Then
        rPup.Value = WorksheetFunction.Max(Me.txtBinomialX - arCell(iCount - 1), 0)
        rPDown.Value = WorksheetFunction.Max(Me.txtBinomialX - arCell(iCount), 0)
    End If

    With rCall

            '12/10/2008
            If Not Me.chkBinomialBSApproximation Then
                .Value = (Me.PFactor * rPup + (1 - Me.PFactor) * rPDown) / (1 + Me.txtBinomialr)
            Else
                .Value = (Me.qU * rPup) + (Me.qD * rPDown)
            End If

             Me.PutPrice = .Value  '12/8/2008

            .Borders(xlBottom).LineStyle = xlContinuous
    End With

    rPDown.Borders(xlBottom).LineStyle = xlContinuous
    With rPup
        .Borders(xlBottom).LineStyle = xlContinuous
        .Offset(1, 0).Resize((rPDown.Row - rPup.Row), 1). _
        Borders(xlEdgeLeft).LineStyle = xlContinuous
    End With

End Sub
Sub CallDecisionTree(rPrice As Range, arCell As Variant, iCount As Long)
    Dim rCall As Range
    Dim rCup As Range
    Dim rCDown As Range

    Set rCall = Me.CallTree.Cells(rPrice.Row, rPrice.Column)
    Set rCup = Me.CallTree.Cells(arCell(iCount - 1).Row, arCell(iCount - 1).Column)
    Set rCDown = Me.CallTree.Cells(arCell(iCount).Row, arCell(iCount).Column)

    If rCup.Column = Me.TreeWorksheet.UsedRange.Columns.Count Then
        With rCup
            .Value = WorksheetFunction.Max(arCell(iCount - 1) - Me.txtBinomialX, 0)
            .Borders(xlBottom).LineStyle = xlContinuous
        End With
        With rCDown
                .Value = WorksheetFunction.Max(arCell(iCount) - Me.txtBinomialX, 0)
                .Borders(xlBottom).LineStyle = xlContinuous
        End With
    End If

    With rCall

        If Not Me.chkBinomialBSApproximation Then
            .Value = (Me.PFactor * rCup + (1 - Me.PFactor) * rCDown) / (1 + Me.txtBinomialr)
        Else
            .Value = (Me.qU * rCup) + (Me.qD * rCDown)
        End If
```

(continued)

```
        Me.CallPrice = .Value  '12/8/2008

        .Borders(xlBottom).LineStyle = xlContinuous
    End With

    rCup.Offset(1, 0).Resize((rCDown.Row - rCup.Row), 1). _
                Borders(xlEdgeLeft).LineStyle = xlContinuous

End Sub

Sub DecitionTreeFormat()
    Dim rTree As Range
    Dim nColumns As Integer
    Dim rLast As Range
    Dim rCell As Range
    Dim lCount As Long
    Dim lCellSize As Long
    Dim vntColumn As Variant
    Dim iCount As Long
    Dim lTimes As Long
    Dim arCell() As Range
    Dim sFormatColumn As String
    Dim rPrice As Range


    Application.StatusBar = "Formatting Tree.. "
    Set rTree = Me.TreeWorksheet.UsedRange
    nColumns = rTree.Columns.Count


    Set rLast = rTree.Columns(nColumns).EntireColumn.SpecialCells(xlCellTypeConstants, 23)
    lCellSize = rLast.Cells.Count
    For lCount = nColumns To 2 Step -1
        sFormatColumn = rLast.Parent.Columns(lCount).EntireColumn.Address
        Application.StatusBar = "Formatting column " & sFormatColumn
        ReDim vntColumn(1 To (rLast.Cells.Count / 2), 1)

        Application.StatusBar = "Assigning values to array for column " & _
            rLast.Parent.Columns(lCount).EntireColumn.Address
        vntColumn = rLast.Offset(0, -1).EntireColumn.Cells(1).Resize(rLast.Cells.Count / 2, 1)
        rLast.Offset(0, -1).EntireColumn.ClearContents

        ReDim arCell(1 To rLast.Cells.Count)
        lTimes = 1
        Application.StatusBar = "Assigning cells to arrays. Total number of cells: " & lCellSize
        For Each rCell In rLast.Cells
            Application.StatusBar = "Array to column " & sFormatColumn & " Cells " & rCell.Row
            Set arCell(lTimes) = rCell
            lTimes = lTimes + 1
        Next

        lTimes = 1

        Application.StatusBar = "Formatting leaves for column " & sFormatColumn
        For iCount = 2 To lCellSize Step 2

            Application.StatusBar = "Formatting leaves for cell " & arCell(iCount).Address
            If rLast.Cells.Count <> 2 Then
                Set rPrice = arCell(iCount).Offset(-1 * ((arCell(iCount).Row - arCell(iCount -
                                1).Row) / 2), -1)
                rPrice.Value = vntColumn(lTimes, 1)
```

(continued)

```
            Else
                Set rPrice = arCell(iCount).Offset(-1 * ((arCell(iCount).Row - arCell(iCount -
                                1).Row) / 2), -1)
                rPrice.Value = vntColumn
            End If


            arCell(iCount).Borders(xlBottom).LineStyle = xlContinuous
            With arCell(iCount - 1)
                .Borders(xlBottom).LineStyle = xlContinuous
                .Offset(1, 0).Resize((arCell(iCount).Row - arCell(iCount - 1).Row), 1). _
                    Borders(xlEdgeLeft).LineStyle = xlContinuous
            End With
            lTimes = 1 + lTimes

            CallDecisionTree rPrice:=rPrice, arCell:=arCell, iCount:=iCount
            PutDecisionTree rPrice:=rPrice, arCell:=arCell, iCount:=iCount
            BondDecisionTree rPrice:=rPrice, arCell:=arCell, iCount:=iCount
        Next

        Set rLast = rTree.Columns(lCount - 1).EntireColumn.SpecialCells(xlCellTypeConstants, 23)
        lCellSize = rLast.Cells.Count
    Next ' / outer next

    rLast.Borders(xlBottom).LineStyle = xlContinuous
    Application.StatusBar = False
End Sub

'/*********************************************************************
'/Purpse: To calculate the price value of every state of the binomial
'/        decision tree
'/*********************************************************************
Sub DecisionTree(rCell As Range, nPeriod As Integer, _
                dblPrice As Double, sngU As Single, sngD As Single)
    Dim lIteminColumn As Long

    If Not nPeriod = 1 Then
        'Do Up
        DecisionTree rCell:=rCell.Offset(0, 1), nPeriod:=nPeriod - 1, _
                dblPrice:=dblPrice * sngU, sngU:=sngU, _
                sngD:=sngD
        'Do Down
        DecisionTree rCell:=rCell.Offset(0, 1), nPeriod:=nPeriod - 1, _
                dblPrice:=dblPrice * sngD, sngU:=sngU, _
                sngD:=sngD
    End If


    lIteminColumn = WorksheetFunction.CountA(rCell.EntireColumn)


    If lIteminColumn = 0 Then
        rCell = dblPrice

    Else
        If nPeriod <> 1 Then
            rCell.EntireColumn.Cells(lIteminColumn + 1) = dblPrice
        Else
            rCell.EntireColumn.Cells(((lIteminColumn + 1) * 2) - 1) = dblPrice
            Application.StatusBar = "The number of binomial calcs are : " & Me.BinomialCalc _
            & " at cell " & rCell.EntireColumn.Cells(((lIteminColumn + 1) * 2) - 1).Address
        End If
```

(continued)

```
    End If
    Me.BinomialCalc = Me.BinomialCalc + 1

End Sub
Function BS_D1() As Double
    Dim dblNumerator As Double
    Dim dblDenominator As Double

    On Error Resume Next
    dblNumerator = VBA.Log(Me.txtBinomialS / Me.txtBinomialX) + _
                  ((Me.txtBinomialr + Me.txtSigma ^ 2 / 2) * Me.txtTimeT)
    dblDenominator = Me.txtSigma * Sqr(Me.txtTimeT)


    BS_D1 = dblNumerator / dblDenominator

End Function
Function BS_D2() As Double

    On Error Resume Next
    BS_D2 = BS_D1 - (Me.txtSigma * VBA.Sqr(Me.txtTimeT))

End Function
Function BS_Call() As Double

    BS_Call = (Me.txtBinomialS * WorksheetFunction.NormSDist(BS_D1)) _
              - Me.txtBinomialX * Exp(-Me.txtBinomialr * Me.txtTimeT) * _
              WorksheetFunction.NormSDist(BS_D2)

End Function
'Used put-call parity theorem to price put option
Function BS_PUT() As Double

    BS_PUT = BS_Call - Me.txtBinomialS + _
             (Me.txtBinomialX * Exp(-Me.txtBinomialr * Me.txtTimeT))
End Function
```