Statistika, Vol. 9 No. 1, 9 – 23
Mei 2009

OPTION VALUATION WITH MATHEMATICA

ONG YIH YING, [1]ANTON ABDULBASAH KAMIL, & [2]MOHAMAD FAISAL ABDUL KARIM

[1,2]School of Distance education, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
email: anton@usm.my, faisal@usm.my

ABSTRACT

Studies in option pricing have became more important and challenging in financial area because option valuation can add significant information to the decision making process. In this study we attempt to establish the American and European call option. Call option gives the buyer the right, but not the obligation to buy a specified stock at a predetermined price on or before a predetermined date. An American option may be exercised earlier before the expiration date. For European options, early exercise is not possible. It can only be exercised only at maturity. In this paper, we do mathematical experiments to solve the problem of financial valuation using Mathematica.
Keywords: Matematica, call option, Binomial option pricing model, American and European Options, Black Scholes option.

1. Introduction

Options are derivative securities. A derivative asset is essentially a financial instrument which its value is dependent on the value of its underlying asset. Options provide the holder the right but not the obligation to buy (for call options) or sell (for put options) a specific amount of an underlying asset at a predetermined price (the exercise price or strike price) during a specified period of time. Each option has a buyer (the holder) and a seller (writer).

The most fundamental difference between options and other financial derivatives is that options provide 'the right but not the obligation' to transact. The owner/buyer does not have to exercise the option if the situation appears to be unprofitable for him/her. Hence, the buyer can let the option expire without using it. The seller, by contrast, has no choice but to buy or sell the financial instrument if the owner chooses to exercise the option. The seller then is responsible for fulfilling the terms of the contract by delivering the options to the appropriate party.

For the owner, the potential loss is limited to the price paid to acquire the option. When an option is not exercised, it expires. So for the buyer, the upside is unlimited whereas for the seller, the potential loss is unlimited.

There are two types of options, the American-style option and the European-style option: (i) American option is an option which can be exercised at any business day up to and including the expiry date (any time during the life of the option). Exercise date is that date on which the option writer accepts valid instructions from the purchaser of the option as to his/her decision to exercise his/her rights under the option. This right to exercise at any time during the life of the option makes the option more expensive, (ii) European option is an option which can only be exercised at the maturity. The restrictiveness of the European option compared to an American option makes it less valuable and thus has a relatively lower premium.

Modern option pricing techniques are often considered among the most mathematically complex of all applied areas of finance. Financial analysts have reached the point where they are able to calculate, with alarming accuracy, the value of a stock option. Most of the models and techniques employed by today's analysts are rooted in a model developed by Fischer Black and Myron Scholes (1973), called The Black and Scholes Option Pricing Model.

Cox, Ross, and Rubinstein (1979), propose the binomial option model as an alternative to the Black-Scholes model. The binomial model can value both European and American options on common stocks that pay dividends, and it explicitly recognizes the possibility of early exercise. It is based on the tendency of a binomial distribution to approach normality as a limit. It is also an easy way to construct price approximations, where no closed form solution is available. The

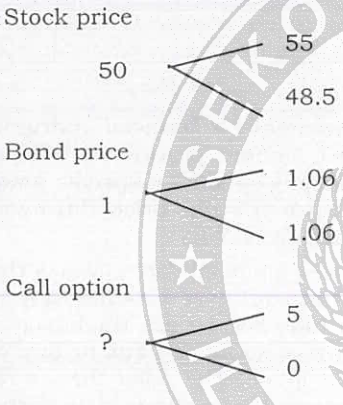
binomial option pricing model is a discrete time model, in that underlying asset price changes at a given fixed time interval.

2. Methodology and Application

2.1 Binomial Option Pricing Model

The price of an option is typically a non-linear function of the underlying asset. The basis of any option pricing model is a description of the stochastic process followed by the underlying asset on which the option is written. Here, we consider the case where the stock price follows a simple, stationary binomial process. At each moment in time, the price can go either up or down by a given percentage. When the stock price follows such a process and when there exists a risk-free asset, options written on the stock are easy to be priced. Furthermore, given the appropriate limiting conditions, the binomial process converges to a lognormal price process and the binomial pricing formula converges to the Black-Scholes formula.

An example: Consider a stock whose price on 30th November is \$50. Suppose that over the next year, the stock price can go either up by 10% or down by -3%, so that the stock price at the end of the year is either \$55 or \$48.50. If there also exists a call on the stock with the exercise price K = \$50, then these three assets will have the following payoff patterns:



In this case the option payoffs can be replicated by a linear combination of the stock and the bond. This combination defines its price uniquely. To see this, denote by A the number of shares and by B the number of bonds which exactly replicate the option's payoffs. This gives the following system of linear equations to solve:

$$55A + 1.06B = 5$$
$$48.5\,A + 1.06B = 0$$

Using *Mathematica* to solve these equations, we get:

```
In[1]:= Solve [{55*A + 1.06*B == 5, 48.5*A + 1.06*B ==0}, {A, B}]
Out[1] = {{A → 0.769231, B → -35.1959}}
```

This system of equations solves to give A = 0.769231, B = -35.1959. Thus purchasing 0.77 of a share of the stock and borrowing \$35.20 at 6% for one period will give payoffs of \$5 if the stock price goes up and \$0 if the stock price goes down the payoffs of the call option. It follows that the price of the option must be equal to the cost of replicating its payoffs, i.e., call option price = 0.7692*\$50 - \$35.1959 = \$3.2656.

This logic is called "pricing by arbitrage": if two assets or sets of assets (the call option and the portfolio of 0.77 of the stock and -\$35.20 of the bonds) have the same payoffs, they must have the same market price. Suppose that the states of "up" and "down" represent all of the

uncertainty in the next period, we can derive market prices for these states by solving the following set of simultaneous equations:

$$p*\$55 + q*\$47.50 = \$50$$

$$p + q = \frac{1}{1.06}$$

The market prices p and q are the price today of an "up" and a "down" state respectively. The price p represents the price today of one dollar in an "up" state tomorrow and the price q is the price today of one dollar in a "down" state tomorrow. The economic logic behind the first equation is that the price paid today for the stock, \$50.00, represents a price paid today for \$55 of stock value in an "up" state tomorrow plus the price paid today for \$47.50 in a "down" state tomorrow. The second equation is derived from much the same logic for the riskless bond: given a price today of \$100, the bond will return \$100*1.06 = \$106 tomorrow in both the "up" and "down" states, so that the relevant pricing equation should be $106*(p + q) = 100$. This system of equation can, of course, be solved using *Mathematica*:

```
In[2]:= Solve [{55*p + 47.5*q == 50, p + q == 1/1.06}, {p, q}]
Out[2] = {{p → 0.691824, q → 0.251572}}
```

Given an "up" and "down" movement, we can easily derive the general expressions for p and q. We can use *Mathematica* to solve the equations, using an upper-case R to denote one-plus-the-interest rate: $R = 1 + r$.

```
In[3]:= a = Simplify [Solve [{p*up + q*down == 1, p + q == 1/R}, {p, q}]]
```

$$Out[3]= \left\{\left\{p \to \frac{down - R}{downR - Rup}, q \to \frac{R - up}{downR - Rup}\right\}\right\}$$

Simplified further:

$$p = \frac{R - down}{R(up - down)}, q = \frac{1}{R} - p$$

Suppose we divide the interval between 0 and 1 into n subintervals. Suppose that in each of these subintervals the price of the stock can go up bu u or down by d and suppose that the riskless interest rate over a subinterval is r. Now let p be the state over one such interval for an up state and q be the state over one such interval for a down state; i.e., $\{p, q\}$ solves

$$p * u + q * d = 1$$

$$p + q = \frac{1}{1 + r}$$

Then the price of the stock at time 1 will be $su^j d^{n-j}$, where s is the initial stock price and j is the number of up movements in the stock price over the period. A European call option with exercise price X and exercise date n will have payoff in state $\{n, j\}$ given by

$$payoff\ in\ state\ (n, j) = Max[su^j d^{n-j} - X, 0]$$

Given the state prices $\{p, q\}$, the option price can be calculated as:

$$C(X) = \sum_{j=0}^{n} p^j q^{j-n} \binom{n}{j} Max\left[su^j d^{n-j} - X, 0\right]$$

Similarly, the price of a European put is given by:

$$P(X) = \sum_{j=0}^{n} p^j q^{n-j} \binom{n}{j} Max\left[X - su^j d^{n-j}, 0\right]$$

Where the binomial coefficient:

$$\binom{n}{j} = \frac{n!}{j!(n-j)!}$$

denotes the number of occurrences of j up states in n subintervals. Implementing this in *Mathematica*:

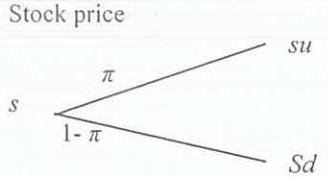
```
In[4]:=   Clear [statePrices]
          statePrices [up_, down_, R_]:=
          Solve [{p*up + q*down == 1, p + q ==1/R},
          {p, q}] [[1]]
          Clear [binomialCall]
          binomialCall [s_, x_, n_]:=
          Sum [p^j*q^(n - j)*Binomial[n, j]*Max[s*up^j*down^(n – j) – x, 0],
          {j, 0, n}] /. statePrices [up, down, R]
          up = 1.1;
          down = 0.97;
          R = 1.06; binomialCall{50, 50, 2]
Out[4]= 5.74917
```

2.2 Determining the Binomial Parameters

Suppose that a stock's return is lognormally distributed with mean μ and standard deviation σ. Denoting the stock price today by s and the stock price at date t in the future by s_t, this means that

$$\log\left(\frac{s_t}{s}\right) \sim N\left(\mu t, \sigma\sqrt{t}\right)$$

We want to determine the parameters of a binomial distribution which, in the limit, will converge to a given lognormal distribution. We first assume that $t = 1$ and that the interval between time *0* and *1* is divided into n subintervals; in each subinterval the stock price can go up or down with a factor u or d; the probability of an increase u is denoted π.



In general all three of these parameters—u, d, and π—will be functions of n. If, at the end of n subperiods, there have been j upward jumps, then the terminal stock price will be

$$su^j d^{n-j}$$

The logarithm of one-plus the return from investing in the stock at date *0* will then be:

$$\log\left[\frac{su^j d^{n-j}}{s}\right] = j\log(u) + (n-j)\log(d) = j\log\left(\frac{u}{d}\right) + n\log(d)$$

Taking the expectation and variance, we get:

$$E\left\{\log\left[\frac{su^j d^{n-j}}{s}\right]\right\} = E(j)\log\left(\frac{u}{d}\right) + n\log(d) = n\pi\log\left(\frac{u}{d}\right) + n\log(d)$$

$$Var\left\{\log\left[\frac{su^j d^{n-j}}{s}\right]\right\} = \text{var}(j)\left[\log\left(\frac{u}{d}\right)\right]^2 = n\pi(1-\pi)\left[\log\left(\frac{u}{d}\right)\right]^2$$

The second equation follows since the variance each period is given by

$$(u-d)^2\left(\pi(1-\pi)^2 + (1-\pi)\pi^2\right) = (u-d)^2\pi(1-\pi)$$

Thus, solving for the parameters u, d, π which converge to a given lognormal distribution involves solving the following equations:

$$n\pi\log\left(\frac{u}{d}\right) + n\log(d) = \mu$$

$$n\pi(1-\pi)\left[\log\left(\frac{u}{d}\right)\right]^2 = \sigma^2$$

Note that there are 3 unknowns and only 2 equations. In solving the equations we can thus almost arbitrarily set one of the unknowns. We do this in *Mathematica*. Below, for example, is the solution when we set $\pi = \_$:

```
In[5]:= a =     Solve [{(pi*Log[u/d] + Log[d])*n == mu,
                n*pi*(1-pi)*Log[u/d]^2 == var} /.
                pi -> ½, {u, d}]
```

$$Out[5]= \left\{\left\{u \to e^{\frac{\mu+\sqrt{n}\sqrt{\text{var}}}{n} - \frac{2\sqrt{\text{var}}}{\sqrt{n}}}, d \to e^{\frac{\mu+\sqrt{n}\sqrt{\text{var}}}{n}}\right\}, \left\{u \to e^{\frac{\mu-\sqrt{n}\sqrt{\text{var}}}{n} + \frac{2\sqrt{\text{var}}}{\sqrt{n}}}, d \to e^{\frac{\mu-\sqrt{n}\sqrt{\text{var}}}{n}}\right\}\right\}$$

As we can see there are two (symmetric) solutions. However, since we want $u>d$, the second solution a[[2]] is the one we are looking for:

```
In[6]:= a[[2]] /. {mu -> 0.12, var -> 0.2^2, p -> 0.5, n -> 100}
Out[6]= {u → 1.02143, d → 0.981376}
```

Substituting in some values and calculating the frequency diagram, we get

```
In[7]:= pi = 0.5;
        mu = 0.12;
        var = 0.2^2;
        n = 100;
        aa = ListPlot[Table[{u^j*d^(n - j),
        pi^j*(1 - pi)^(n - j)*Binomial[n, j]},
        {j, 0, n}] /. a[[2]],
        PlotJoined -> True,
        PlotRange -> All];
```

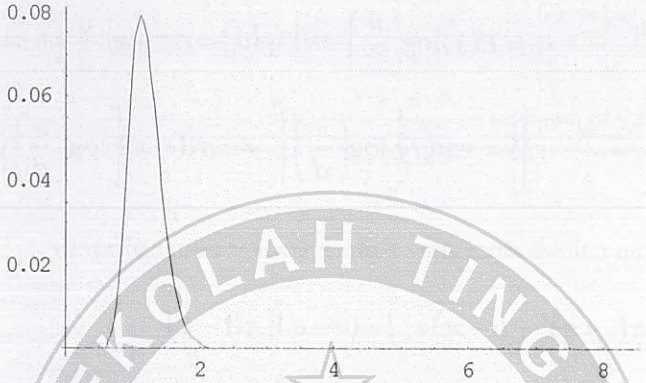


Figure 1: Frequency Diagram

Note that this is not the only set of values which converges to the lognormal. Cox, Ross, Rubinstein (1979), use the following values:

$$u = e^{\sigma\sqrt{1/n}}, d = \frac{1}{u} = e^{-\sigma\sqrt{1/n}}, \pi = \frac{1}{2} + \frac{\mu}{2\sigma}\sqrt{1/n}$$

Note that these values do not solve the equations precisely, they converge to the correct values as $n \to \infty$. We can see this convergence using *Mathematica*:

```
In[8]:= Clear[q, u, d, pi, mu, sigma, n]
        mu = 0.12;
        sigma = 0.2; n = 10;
        b = {pi -> 1/2*(1 + mu/sigma*Sqrt[1/n]),
        u -> E^(sigma*Sqrt[1/n]),
        d -> E^(-sigma*Sqrt[1/n])}
        bb = ListPlot[Table[{u^j*d^(n - j),
        pi^j*(1 - pi)^(n - j)*Binomial[n, j]},
        {j, 0, n}] /. b, PlotRange -> All,
        PlotStyle -> PointSize[0.02]];
        {π→0.594868,u →1.06529,d →0.938713}
```

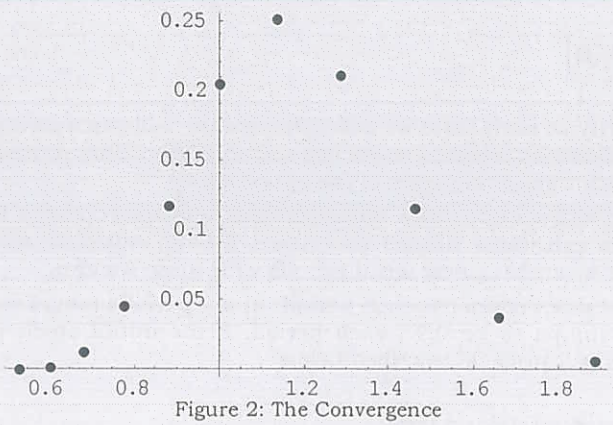


Figure 2: The Convergence

Graphing the previous graph (when $\pi = \frac{1}{2}$) with this graph:

In[9]:= Show[aa, bb];

Figure 3: The Combination

When we have $n = 100$, these two graphs are indeed very close:

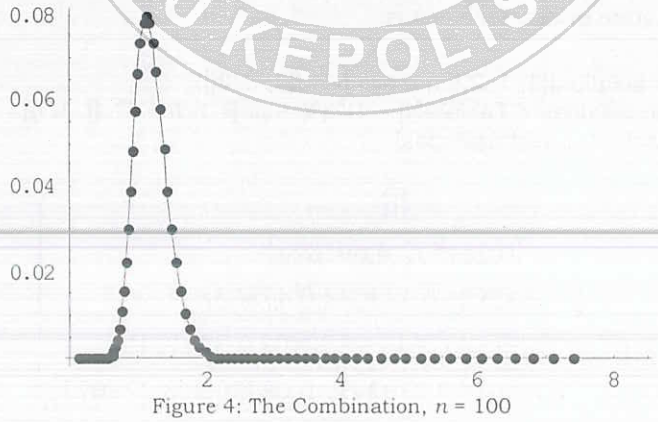


Figure 4: The Combination, $n = 100$

2.3 European Binomial Option Prices Without Dividends

Suppose a single period is divided into T intervals, so that the price of the stock at the end of the last interval can be written as S_T. Then the call option price at maturity coincides with its payoff function,

$$Max[s_T - X, 0]$$

A day prior to maturity at each state we can calculate the call price as a weighted average (with risk-neutral probabilities as weights) of its price at maturity. This procedure can be continued to the root of the stock tree, giving price of the option today.

The above procedure will work for both European and American options. However, in the case of European options we can use a simpler procedure. We can calculate state prices of each state at maturity and take a weighted average of payoffs with these weights.

Assume that interest rates equal 6% each period. In a 4 periods model we expect "up" jumps to be 1.1 and "down" jumps to be 0.95 each period. If the initial stock price is $50, then the development of the stock price is described below:

```
In[10]:= up = 1.1; down = 0.95; r = 0.06;
         stock = Table[50*up^(j - 1)*down^(i - j), {i, 1, 5}, {j, 1, i}];
         MatrixForm[stock]
```

Out[10]=

$$\begin{pmatrix} \{50\} \\ \{47.5, 55.\} \\ \{45.125, 52.25, 60.5\} \\ \{42.8687, 49.6375, 57.475, 66.55\} \\ \{40.7253, 47.1556, 54.6013, 63.2225, 73.205\} \end{pmatrix}$$

Note that the indices go from 1 to $n + 1$ and from 1 to i, which means that the date 0 corresponds to the node with indices (1,1) and the last date all up nodes corresponds to indices $(n + 1, n + 1)$.

To find one period state prices,

```
In[11]:= solution = Solve[{p*(1 + r) + q*(1 + r) == 1, up*p + down*q == 1}, {p, q}]
Out[11]= {{p → 0.691824, q → 0.251572}}
```

Note that r is the interest rate for each period, not the annual interest rate.

The whole tree of state prices, for $n = 4$ is:

```
In[12]:= n = 4; p = solution[[1, 1, 2]]; q = solution [[1, 2, 2]];
         statePrices = Table[p^(j – 1)*q^(i – j), {i, 1, n + 1}, {j, 1, i}];
         MatrixForm[statePrices]
```

Out[12]=

$$\begin{pmatrix} \{1\} \\ \{0.251572, 0.691824\} \\ \{0.0632886, 0.174044, 0.47862\} \\ \{0.0159217, 0.0437846, 0.120408, 0.331121\} \\ \{0.00400545, 0.011015, 0.0302912, 0.0833009, 0.229077\} \end{pmatrix}$$

Define the payoff function for the European call and put options with exercise price X as:

```
In[13]:= Clear[payoffcall, payoffPut];
         payoffCall[s_]:= Max[s – X, 0];
```

```
payoffPut[s_]:= Max[X – s, 0];
```

To find the price today of the option, we sum all possible final payoffs at maturity $(n + 1)$ with weights given by the corresponding state prices. Recall that the state price today of each final node of the tree is exactly the price one should pay for \$1 received if and only if this state is realized.

```
In[14]:= X = 45;
         statePrices[[n + 1]] . payoffCall /@stock[[n + 1]]
Out[14]= 8.29366
```

By doing this, we can apply the function payoff Call to the whole list of stock prices at the final day. This gives the list of final payoffs which we multiply it by the vector (list) of the corresponding state prices. The result is the price today of an European call on the tree.

This method is very useful when the price today is the only variable of interest and nothing important happens in the intermediate dates (such as dividends, changes in volatility or interest rates, early exercise, etc.).

Another way to calculate the price of a European option is to build a tree with all intermediate prices. This gives much more flexibility since one can introduce different events in any intermediate date.

However, the approach described above is very transparent but not very efficient. There is a more efficient way which uses the same algorithm. Here we do not have to guess the sizes of each period up and down jumps. Instead we use the annual historical volatility and interest rates.

```
In[15]:=Clear[up, down, R, P, Q, EuropeanOption, EuropeanCall, EuropeanPut, mean];
        up[n_, sigma_, T_]:= N[Exp[Sqrt[T/n]*sigma]];
        down[n_, Rf_, T_]:= 1/up[n, sigma, T];
        R[n_, Rf_, T_]:= N[Exp[Rf*T/n]];
        P[up_, down_, r_]:= N[(r – down)/(up – down)/r];
        Q[up_, down_, r_]:= N[1/r – P[up, down, r]];
        mean[m_List]:= Plus @@ m/Length[m];
```

Using these definitions we can write a simple function which calculates the price of a European option:

```
In[16]:= EuropeanOption[s_, sigma_, T_, Rf_, exercise_Function, n_]:=
         Module[{   u = up[n, sigma, T],
                    d = down[n, sigma, T],
                    r = R[n, Rf, T], p, q},
                    p = P[u, d, r];
                    q = Q[u, d, r];
         Sum[exercise[s*u^j*d^(n – j)]*Binomial[n, j]*p^j*q^(n – j), {j, 0, n}]];
```

We are able to give the precise formula for state prices because of the assumption that everything is stationary (constant volatility, interest rates, etc.). The two most popular options—call and put can be calculated as follows:

```
In[17]:=   EuropeanCall[s_, X_, sigma_, T_, Rf_, n_]:= EuropeanOption[s, sigma, T, Rf,
           Max[# - X, 0]&, n];
           EuropeanPut[s_, X_, sigma_, T_, Rf_, n_]:= EuropeanOption[s, sigma, T, Rf,
           Max[X - #, 0]&, n];
```

This is the definition of the European option with the payoff at exercise functions corresponding to a call and put. # is used for an argument of a pure function and the & sign is used to define a pure function. For example we can calculate the values of the following two options, both of which have $s = 50$, $X = 45$, $\sigma = 40\%$, $T = 1$, and $r = 10\%$. In both cases we divide the time T into 100 subintervals:

```
In[18]:= EuropeanCall[50, 45, 0.4, 1, 0.1, 100]
         EuropeanPut[50, 45, 0.4, 1, 0.1, 100]
Out[18]= 12.7526
         3.47028
```

2.4 American Options Without Dividends

The pricing of American options is different than of European options because of the early exercise option. The answer of when it is optimal to exercise an American option is very simple. We know that the price of the option if it is alive at maturity it is given by the option payoff function. A day before the final date we have a dilemma of whether to exercise the option or not. By exercising early we obtain the option payoff given the current stock price, and by leaving the option alive we continue to hold an option whose value at the end of the next day is its price in the up and down states tomorrow. Denoting the state prices by p and q, the value of the unexercised option one day before the terminal date is:

Option payoff (next period, up)*p + option payoff (next period, down)*q

This should be compared to the value of the option if exercised one day before the terminal date:

Option payoff (the current stock price).

The exercise decision depends on which of these two values is higher.

The tree of stock prices and state prices are the same. Now we calculate all intermediate values of the option in order to decide whether to exercise it prior to maturity.

```
In[19]:= AO = Table[0, {i, 1, n + 1}, {j, 1, i}];
```

This is a list of lists of lengths 1, 2, ..., $n + 1$ corresponding to one node at the root, two nodes after one period, etc. To assign values at maturity one can use either

```
In[20]:= AO[[n + 1]] = Table[payoffCall[stock[[n + 1, j]]], {j, 1, n + 1}];
```

or equivalently:

```
In[21]:= AO[[n + 1]] = payoffCall /@ stock[[n + 1]];
```

Now we can run a backward induction pricing American option by choosing between a live option and its intrinsic value:

```
In[22]:= For[nn = n, nn >= 1, nn-,
           For[j = 1, j <= nn, j++, AO[[nn, j]] = Max[payoffCall[stock[[nn, j]]], p*AO[[nn + 1, j
           + 1]] + q*AO[[nn + 1, j]]]]];
```

Finally in the cell AO[[1,1]] we find the price today of the option.

Another example:

```
In[23] := AmericanOption[s_, n_, sigma_, T_, Rf_, exercise_Function]:=
          Module[{   u = up[n, sigma, T],
                     d = down[n, sigma, T],
                     r = R[n, Rf, T], p, q, OpRecurse},
                     p = P[u, d, r]; q= Q[u, d, r];
          OpRecurse[node_, level_]:= OpRecurse[node, level] = If[level == n,
          exercise[s*d^node*u^(level – node)],
          Max[{p, q} . {OpRecurse[node, level + 1],
          OpRecurse[node +1, level + 1]},
          exercise[s*d^node*u^(level – node)]]];
          OpRecurse[0, 0]];
```

Here we define American Option as a function of the same variables as before. The difference begins with recursive definition of the option price. At level n we use the exercise price as definition, at all other levels one should choose the maximum between option weighted sum of the next period prices and its intrinsic value which is equal to the final payoff calculated at the current stock value.

We can now define American call and put options:

```
In[24]:=   AmericanCall[X_, s_, n_, sigma_, T_, Rf_]:= AmericanOption[s, n, sigma, T, Rf,
           Max[#1 – X, 0] & ];
           AmericanPut[X_, s_, n_, sigma_, T_, Rf_]:= AmericanOption[s, n, sigma, mmT,
           Rf, Max[X - #1, 0] & ];
```

To calculate prices of options we can try:

```
In[25]:= AmericanCall[50, 50, 30, 0.4, 0.5, 0.1]
         AmericanPut[50, 50, 30, 0.4, 0.5, 0.1]
Out[25]= 6.74401
         4.58748
```

2.5 The Black-Scholes Option Pricing Formula

Black-Scholes (1973) proved the following theorem:

Consider a European call option on a stock whose current price is S. Suppose that the stock price is lognormally distributed with volatility σ, that the option's exercise price is X, that the exercise date of the option is T, and that the continuously compounded interest rate is r. Furthermore assume that the stock will pay no dividends before the option exercise date T. Then the call price is given by:

$$C = SN(d_1) - Xe^{-rT} N(d_2),$$

where

$$d_1 = \frac{\ln\left(\frac{s}{x}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \qquad \text{and} \qquad d_2 = d_1 - \sigma\sqrt{T}$$

and where N(.) indicates values of the cumulative standard normal distribution.

By the Put-Call Parity Theorem, suppose we are pricing a European call and put, both with exercise date T and both defined on a stock whose current price is S and which does not pay dividends before the maturity date T. Then if the continuously compounded interest rate is r, the value P of the put, the value C of the call, and the stock price S are related

by $P = C + Xe^{-rT} - S$. Applying this theorem, it follows that the Black-Scholes price of a European put written on the same stock with the same exercise price and date is given by $P = -SN(-d_1) + Xe^{-rT}N(-d_2)$.

Four solutions are suggested to implement the Black-Scholes formula in *Mathematica*. The four functions snormal are different in speed of computation and other properties.

```
In[1]:= snormal1[x_]:= Integrate[Exp[-z^2/2]/Sqrt[2*Pi], {z, -Infinity, x}]//N
        snormal2[x_]:= NIntegrate[Exp[-z^2/2]/Sqrt[2*Pi], {z, -Infinity, x}];
        snormal3[x_]:= Erf[x/Sqrt[2]]/2 + 0.5;
In[2]:= Needs["Statistics`Master`"]
        ndist = NormalDistribution[0, 1];
In[3]:= snormal4[x_]:= CDF[ndist, x]//N;
```

To compare performance, we can calculate 100 values of the normal distribution for each function:

```
In[4]:= Timing[Table[snormal1[i/100], {i, 100}]; ][[1]]
        Timing[Table[snormal2[i/100], {i, 100}]; ][[1]]
        Timing[Table[snormal3[i/100], {i, 100}]; ][[1]]
        Timing[Table[snormal4[i/100], {i, 100}]; ][[1]]
Out[4]= 12.328 Second
        0.156 Second
        0. Second
        0.016 Second
```

The results show how much faster the calculation can be performed with built-in functions (the fourth definition is a lot faster than the first). Having defined the normal distribution, we program the Black-Scholes formula for both puts and calls:

```
In[5]:= Clear[snormal, d1, d2, bsCall, bsPut]
        snormal = snormal3;
        d1[s_, x_, sigma_, T_, r_]:=
            (Log[s/x] + (r + sigma^2/2)*T)/(sigma*Sqrt[T])
        d2[s_, x_, sigma_, T_, r_]:=
            d1[s, x, sigma, T, r] – sigma*Sqrt[T]
        bsCall[s_, x_, sigma_, T_, r_]:=
            s*snormal[d1[s, x, sigma, T, r]]-x*
            Exp[-r*T]*snormal[d2[s, x, sigma, T, r]]
        bsPut[s_, x_, sigma_, T_, r_]:=
            bsCall[s, x, sigma, T, r] + x*Exp[-r*T] – s
```

We can use *Mathematica* to graph these functions:

```
In[6]:= Plot[bsCall[s, 50, 0.2, 0.5, 0.05], {s, 0.01, 70}, PlotRange -> All, AxexLabel -> {"stock
        price", "call price"}]
```

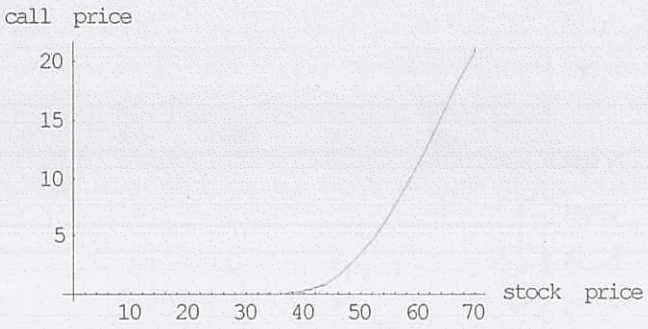


Figure 5: Call Price vs. Stock Price

2.6 The Binomial Option Price Converges to the Black-Scholes Price

Previously, we defined a binomial option pricing formula for European options for which—in the limit—the stock's price process converges to the lognormal price:

```
In[7]:= Clear [EuropeanOption, EuropeanCall, EuropeanPut]
        EuropeanOption[s_, sigma_, T_, r_, exercise_Function, n_]:=
            Module[{
                u = N[Exp[Sqrt[T/n]*sigma]],
                d = N[Exp[-Sqrt[T/n]*sigma]],
                R = N[Exp[r*T/n]]},
                p = (R – d)/(R*(u – d));
                q = (u – R)/(R*(u – d));
            Sum[exercise[s*u^j*d^(n – j)]*
                Binomial[n, j]*p^j*q^(n – j), {j, 0, n}]]
        EuropeanCall[s_, x_, sigma_, T_, r_, n_]:=
            EuropeanCall[s, x, sigma, T, r, n] =
                EuropeanOption[s, sigma, T, r, Max[#1 – x, 0] &, n];
        EuropeanPut[s_, x_, sigma_, T_, r_, n_]:=
            EuropeanPut[s, x, sigma, T, r, n] =
                EuropeanOption[s, sigma, T, r, Max[x - #1, 0] &, n];
```

We can now show that this binomial pricing formula converges to the Black-Scholes price:

```
In[8]:= ListPlot[Table[{n, bsCall[50, 45, 0.4, 0.25, 0.06] –
        EuropeanCall[50, 45, 0.4, 0.25, 0.06, n]}, {n, 10, 500, 10}],
        PlotJoined -> True, PlotRange -> All, AxesLabel -> {n, ""}];
```

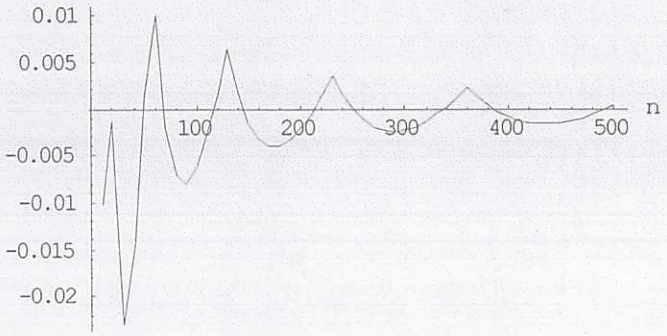


Figure 6: Convergence of Binomial Option Pricing model to the Black-Scholes

We also defined the following function to price American options:

```
In[9]:= AmericanOption[s_, sigma_, T_, r_, exercise_Function, n_]:=
          Module[{u = N[Exp[Sqrt[T/n]*sigma]],
                  d = N[Exp[-Sqrt[T/n]*sigma]],
                  R = N[Exp[r*T/n]], p, q, OpRecurse},
                  p = (R – d)/(R*(u – d)); q = (u – R)/(R*(u – d));
        OpRecurse[node_, level_]:=
          OpRecurse[node, level] = If[level == n,
                  exercise[s*d^node*u^(level – node)],
        Max[{p, q} . {OpRecurse[node, level + 1],
          OpRecurse[node + 1, level + 1]},
          exercise[s*d^node*u^(level – node)]]];
        OpRecurse[0, 0]];
        AmericanCall[s_, x_, sigma_, T_, r_, n_]:=
          AmericanCall[s, x, sigma, T, r, n] =
          AmericanOption[s, sigma, T, r, Max[# - x, 0]&, n];
        AmericanPut[s_, x_, sigma_, T_, r_, n_]:=
          AmericanPut[s, x, sigma, T, r, n] =
          AmericanOption[s, sigma, T, r, Max[x - #, 0]&, n];
```

The price of an American call on a stock without dividends (as in the case here) is the same as the price of a European option. However, it need not be that an American put and a European put have the same price. Both of these properties are illustrated below:

```
In[10]:= Table[{n, AmericanCall[50, 45, 0.4, 0.25, 0.06, n] –
           EuropeanPut[50, 45, 0.4, 0.25, 0.06, n]}, {n, 1, 100, 10}]
In[11]:= a = Table[{n, AmericanPut[50, 45, 0.4, 1, 0.10, n] –
           EuropeanPut[50, 45, 0.4, 1, 0.10, n]}, {n, 10, 80, 10}];
In[12]:= ListPlot[a, PlotStyle -> PointSize[0.02], Frame -> {True, True, False, False},
           FrameLabel -> {"n = iterations", difference},
           PlotLabel -> "Difference Between American and European Put",
           DefaultFont -> {"Helvetica", 9}];
```

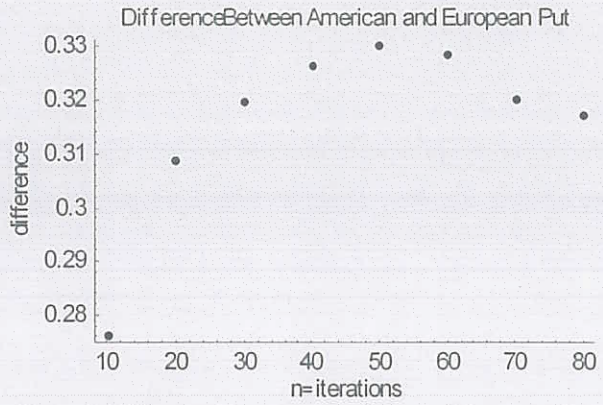


Figure 7: Difference Between American and European Put

It follows that the early exercise option of the American call is worthless. However, the early exercise feature of American puts is potentially valuable.

3. Conclusion

According to the binomial method, we can calculate at each state the European call price as a weighted average (with risk-neutral probabilities as weights) of its price at maturity a day prior to maturity. This can be continued to the root of the stock tree, giving price of the option today. Alternatively, we can use the annual historical volatility and interest rates instead of guessing the size of each period up and down jumps. Whereas for American options, the intermediate values of the option are calculated to decide whether to exercise it prior to maturity. Then a backward induction is conducted.

The Black-Scholes model on the other hand values a European call assuming the stock price to be log normally distributed with compounded interest rate and that the stock pays no dividends before the option exercise date.

It is shown that the binomial pricing formula converges to the Black-Scholes model as the number of subintervals increase.

Acknowledgements

The work funded by the Short Term Research Project, Universiti Sains Malaysia, Grant 304/PJJAH/638072.

References

[1]. Black, Fischer and Myron Scholes. 1973. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy,* 81, 637-659.

[2]. Cox, John C., and Mark Rubinstein. 1985. *Options Markets.* Englewood Cliffs, NJ: Prentice-Hall.

[3]. Cox, J. C., Ross, S. A., and Rubinstein, M. 1979. *Option pricing: a simplified approach. Journal of Financial Economics,* 7:229-263.

[4]. DeRosa, David F. 2000. *Options on Foreign Exchange,* 2nd edition. New York: John Wiley & Son, Inc.

[5]. Miller, Ross M. 1990. Computer-Aided Financial Analysis: An Implementation of the Black-Scholes Model. *Mathematica Journal,* 1, 75-79.

[6]. Varian, H.R. et all. 1993. *Economic and Financial Modeling with Mathematica.* New York: Springer-Verlag, Inc.


SEKOLAH TINGGI
ILMU KEPOLISIAN
BHAKTI - DHARMA - WASPADA