# The Binomial Option Pricing Model

*The authors consider the case of option pricing for a binomial process—the first in a series of articles in Financial Engineering.*

**by Simon Benninga and Zvi Wiener**

The two major types of securities are stocks and bonds. A share of stock represents partial ownership of a company with an uncertain payoff which depends on the success of the specific business. Bonds, on the other hand, are loans which must be repaid except in cases of default; they can be issued either by governments or by business.

Modern financial markets offer many other instruments besides stocks and bonds. Some of these instruments are called derivatives, since their price is derived from the values of other assets. The most popular example of a derivative is an option, which represents a contract allowing to one side to buy (in the case of a call option) or to sell (the case of a put option) a security on or before some specified maturity date in the future for a price which is set today. This prespecified price is called the exercise price or the strike price of the option.

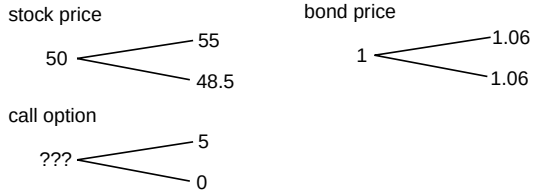
The two major classes of options are called European and American. A European option can be exercised only at maturity while an American option can be exercised at any time prior to maturity.

Option pricing is topic of the current and the following article in this series. The price of an option is typically a non-linear function of the underlying asset (and some other variables, like interest rates, strike, etc.) 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. In the Black-Scholes model, which will be discussed in the next article, the assumption is that the stock price is lognormally distributed (that the price follows a Wiener process with constant drift and variance).

In this article 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 price. Furthermore, given 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 today 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 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:

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

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}}
```

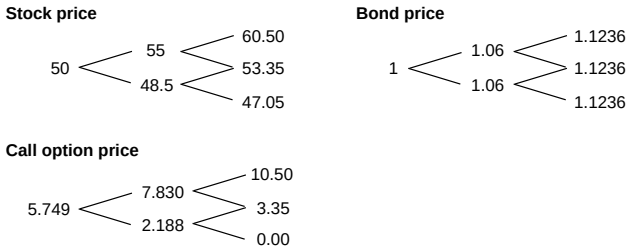
This system of equations solves to give $A = 0.769231$, $B = -35.1959$. Thus purchasing 0.77 of a share of the stock and borrowing \$35.20 at 6% for one period will give payoffs of \$5 if the stock price goes up and \$0 if the stock price goes down—the payoffs of the call option. It follows that the 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 (in our case—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.

## A TWO-DATE EXAMPLE

We can extend this example to multiple periods. Here, for example, is a two-date example; the price of a call option

with exercise price 50 solves (as indicated) to 5.749.




The solution of this problem with *Mathematica* is left as an exercise.

## STATE PRICES AND THE BINOMIAL OPTION PRICING MODEL

We now reconsider the first example of the previous section. Suppose that the *states of the world* called "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:

$$p * \$55 + q * \$47.50 = \$50.00$$
$$p + q = \frac{1}{1.06}$$

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 $q$ 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 the "down" states, so that the relevant pricing equation should be $106∗(p+q) = 100$. Hence the second equation.

This system of equations 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 $q$. As an amusement we will 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}]]
```

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

A little algebra will show that this solution can be simplified further:

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

We can make this example more complicated. 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 by $u$ or down by $d$ and suppose that the riskless interest rate over a subinterval is $r$. Now let $p$ be the state price over one such interval for an up state and $q$ be the one-interval state price; 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:

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

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

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

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* is simple:

```
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
```

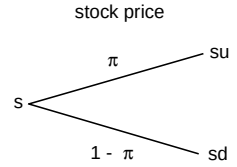
What is still lacking is the connection between the binomial price process discussed in this section and the lognormal price process which underlies the Black-Scholes formula. We discuss this connection in the next section.

## DETERMINING THE BINOMIAL PARAMETERS

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

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

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 time *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 $\pi$.




In general all three of these parameters—$u$, $d$, and $\pi$—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:

$$E\left\{\log\left[\frac{su^j d^{n-j}}{s}\right]\right\} =$$

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

$$Var\left\{\log\left[\frac{s\,u^i\,d^{n-j}}{s}\right]\right\} =$$

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

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

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

Thus, solving for the parameters $u$, $d$, $\pi$ which converge to a given lognormal distribution involves solving the following two 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 two equations; in solving the equations we can thus arbitrarily (almost ...) 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 -> 1/2, {u, d}]
```

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

(**Note:** These equations will solve in *Mathematica* version 2 only if you append `//PowerExpand` to the equations)

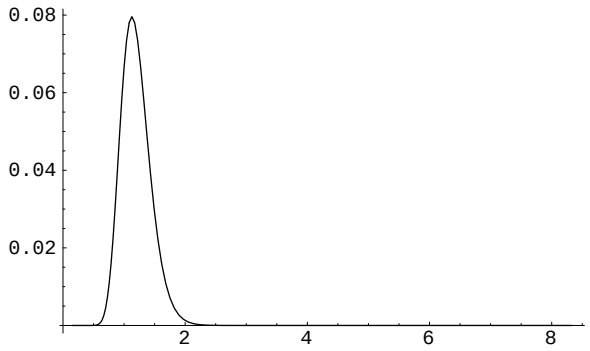
As you can see, there are two (symmetric) solutions. However, since we want $u > d$, the second solution `a[[2]]` is the one we are looking for:

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

$$\text{Out[6]} = \{u \to 1.02143,\ d \to 0.981376\}$$

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];
```



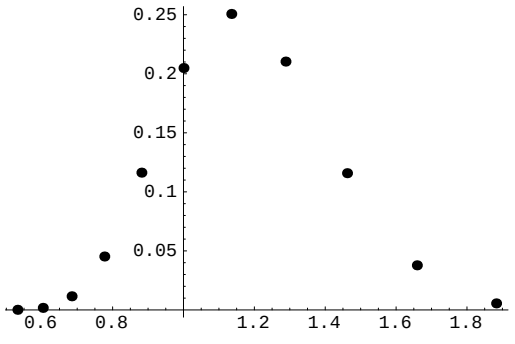

Note that this is not the only set of values which converges to the lognormal. [Cox, Ross, Rubinstein 1979] use the following values:

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

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

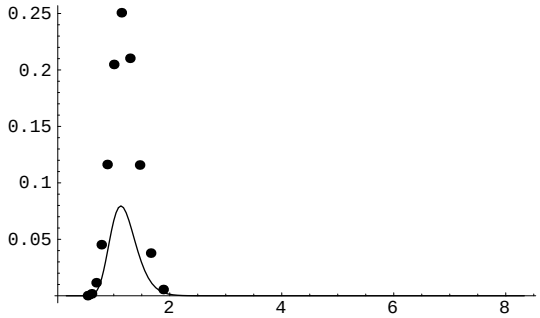
```
In[8]:= Clear[q, u, d, pi, mu, sigma, n]
        mu = 0.12;
        sigma = 0.2; n = 10;
          b = {pi -> 1/2*(1 + mu/sigma*Sqrt[1/n]),
          u -> E^(sigma*Sqrt[1/n]),
            d -> E^(-sigma*Sqrt[1/n])}
        bb = ListPlot[Table[{u^j*d^(n - j),
        pi^j*(1 - pi)^(n - j)*Binomial[n,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

$$\{\pi \to 0.594868, u \to 1.06529, d \to 0.938713\}$$

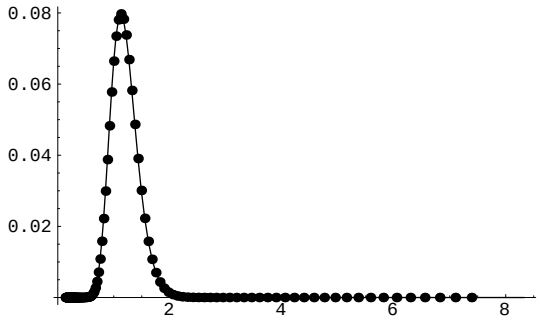



Graphing the previous graph (when $\pi = \frac{1}{2}$) with this graph:

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




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




## EQUIVALENT MARTINGALE MEASURES: RISK-NEUTRAL PRICING

The state prices

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

probabilities, since $p + q = 1/R$. Multiplying the state prices by $R = 1 + r$ allows us to regard them as "pseudo probabilities". The technical name for this is "equivalent martingale measure." Since we are about to price options (and other derivative securities) by using state prices, we could also regard the pricing in an equivalent fashion: Instead of pricing by $p$ and $q$, we can price by $Rp$ and $Rq$. This allows us to say:

"In a perfect market, there exists an equivalent probability measure such that the price of any security is its discounted expected value with respect to this measure."

In more complicated situations, the equivalent probability measure is given by the Radon-Nikodym derivative. For the simple binomial case discussed here, we can easily see the application of this approach. Take the stock price, for example: After one period, the stock price is either $su$ or $sd$. Taking the expected value using the equivalent probability weights $\{pR, qR\}$ gives

$$pR \cdot su + qR \cdot sd = Rs$$

so that the discounted value of the expected value is indeed the stock price:

$$\frac{1}{R} \cdot [pR \cdot su + qR \cdot sd] = s$$

Of course, given our definition of "up" and "down" movements and given the interest rate $r$, this is tautological, since our definition of $p$ and $q$ is derived in such a way as to make this true. Nevertheless, we can prove the statement for any market in which pricing does not allow arbitrage opportunities.

Furthermore, we can—in a non-trivial way—use the "risk-neutral" pricing principle (i.e., pricing returns by discounting their expected return at a properly defined equivalent probability measure) to value options. Thus, for example, if in a binomial world a security pays off $X$ in an "up" state and $Y$ in a "down" state, then its market value today will be given by:

$$value = \frac{1}{R} \cdot [pR \cdot X + qR \cdot Y]$$

## EUROPEAN BINOMIAL OPTION PRICES WITHOUT DIVIDENDS

In this section we develop the binomial pricing model for options. We start with a simple European call option. 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 $s_T$. 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. Obviously 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 based on observation that only the distribution of payoffs at maturity

matters. We can calculate state prices of each state at maturity and take a weighted average of payoffs with these weights.

*Example 1.* 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]=
{{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}}
```

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 ups node corresponds to indices $(n+1, n+1)$. The *Mathematica* convention is not to use zero as an index of an array; for example, we could have written the stock prices in period 4 by using the list `junk` produced by `junk = Table[50*up^(j-1)*down^4-j), {j,0,4}]`. This would have produced the same output as above:

```
{40.7253, 47.1556, 54.6012, 63.2225, 73.205}
```

However (and this is the important part!), *Mathematica* will not recognize `junk[[0]]` as the first item in this list; for *Mathematica* the list `junk` is indexed from 1 to 5, not from 0 to 4. Thus the first index here is the actual day + 1 and the second index is the number of "up" jumps + 1. We show a way to avoid this inconvenience later.

To find one period state prices we use the method described above.

```
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 the case $n$=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]=
{{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}}
```

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
```

Note that the operator `Map` allows us to apply the function `payoffCall` to the whole list of stock prices at the final day. This gives the list of final payoffs which we multiply (dot is a *Mathematica* operator for scalar product of vectors) it by the vector (list) of the corresponding state prices. The result is the price today of a `EuropeanCall` option 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 (like dividends, splits, 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. We consider examples of this approach later.

The approach described above is very transparent but not very efficient. We now show 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_, sigma_, 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}]];
```

The function `EuropeanOption` takes the current stock price `s`, number of periods $n$, the annualized volatility `sigma`, the time to maturity `T` (in years), the annual risk free interest rate `Rf`, and the `exercise` function as arguments. The internal variables $u$, `d`, `r`, `p` and `q` are defined in a transparent way. The last operator takes the sum over all possible final states of payoff with state prices used as weights. We are able to give the precise formula for state prices because of the assumption that everything is stationary (constant volatility, interest rates, etc). This was a description of a generic European type option. The two most popular options—call and put can be calculated as follows:

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

However the *Mathematica* pure function `Max[#-X,0]&` allows to us do this in a shorter way:

```
In[18]:=
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 exactly the definition of the European option with the payoff at exercise functions corresponding to a call and a put. Note that the # 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[19]:= EuropeanCall[50, 45, 0.4, 1, 0.1, 100]
         EuropeanPut[50, 45, 0.4, 1, 0.1, 100]

Out[19]= 12.7526

         3.47028
```

## AMERICAN OPTIONS WITHOUT DIVIDENDS

The pricing of American options differs from that of European options because of the early exercise option. When it is optimal to exercise an American option? In principle the answer is very simple. We know the price of the option if it is alive at the final day—it is given by the option payoff function. A day before the final date we have a dilemma whether to exercise the option or to wait. 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.

### Simple Model

Again we present first a transparent but not very efficient model and then move to more object oriented functional approach. The tree of stock prices and state prices are the same. But now we have to calculate all intermediate values of the option in order to decide whether to exercise it prior to maturity.

Let's prepare a table for values of the American option:

```
In[20]:= AO = Table[0, {i, 1, n + 1}, {j, 1, i}];
```

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

```
In[21]:= AO[[n + 1]] =
           Table[payoffCall[stock[[n + 1,j]]],
             {j, 1, n + 1}];
```

or equivalently:

```
In[22]:= 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[23]:= 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.

### Structural Approach

Here is a more sophisticated example that is essentially based on the same algorithm but uses a very powerful tool—recursion.

```
In[24]:=
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 first define AmericanOption as a function of the same variables as before. The difference begins with OpRecurse—recursive definition of the option price.

Here we define recursively the price. At level $n$ we use the exercise price as definition, at all other levels one should choose the maximum between option alive (weighted sum of the next period prices that are already calculated) and its intrinsic value which is equal to the final payoff calculated at the current stock value.

After the general definition of the AmericanOption we can define the two widely used types of derivative securities—AmericanCall and AmericanPut options.

```
In[25]:=
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] & ];
```

This definition is similar to the one for European options. To calculate prices of options try:

```
In[26]:= AmericanCall[50, 50, 30, 0.4, 0.5, 0.1]
         AmericanPut[50, 50, 30, 0.4, 0.5, 0.1]
Out[26]= 6.74401
         4.58748
```

Our next article will continue the discussion of binomial option pricing models. We will show how to price options when the underlying security pays dividends, how to determine the optimal exercise boundary, and how to price exotic (path dependent) options.

## A COURSE IN FINANCIAL ENGINEERING

For the past several years we have been teaching a course in "Financial Engineering" at the Hebrew University in Israel. The course covers advanced financial programming and uses *Mathematica* as its primary computing and teaching vehicle. Students in the course include both undergraduates and MBAs; our prerequisites are that all students have some prior finance courses (an introductory options course is the minimum requirement) and some "mathematical sophistication"—meaning that they do not startle when confronted by a matrix or an integral sign. The course has been very successful. Students enjoy learning how to implement computationally intricate finance models. As might be suspected, computation turns out to be not merely a goal, but leads to a greater understanding of the finance models themselves. This is the first of a series of articles based on our course. The articles cover a variety of topics in option and bond pricing; their order and level corresponds roughly to the first 10 weeks of a semester. While we will explain the advanced finance topics covered in the articles, we will assume some acquaintance with basics. The subjects we will cover will include:

- Binomial option pricing models
- Black-Scholes option pricing
- Simulating stock prices and dynamic hedging strategies
- Risk management
- Portfolio insurance strategies
- Term structure models

In planning and implementing the course, we have benefited from a series of grants from the Krueger Center for Finance and Banking of the Hebrew University.

### REFERENCES

BENNINGA, S., *Financial Modeling*. MIT Press (1997).

BENNINGA, S., STEINMETZ, R., STROUGHAIR, J., "Implementing Numerical Option Pricing Models", *Mathematica Journal* 3:4, pp. 66–73, 1993.

BERGMAN, Y., GRUNDY, B., WEINER, Z., "Generalized Theory of Rational Option Pricing," *Journal of Finance* 51:5, pp. 1573–1610, 1996.

COX, J. C., ROSS, S. A., RUBINSTEIN, M. (1979). "Option Pricing: A Simplified Approach," *Journal of Financial Economics* 7, pp. 229-263.

HULL J., *Options, Futures, and Other Derivatives*, third edition, Prentice Hall, 1997.

OMBERG, E., "A Note On The Convergence Of Binomial-Pricing And Compound-Option Models," *Journal of Finance*, 1987, v42(2), 463-469.

### ABOUT THE AUTHORS

Simon Benninga is a professor of finance at Tel-Aviv University (Israel) and the Wharton School of the University of Pennsylvania. He has published academic papers in many areas of finance. His newest book, *Financial Modeling*, will be published by MIT Press in October 1997. He is the author of *Numerical Techniques in Finance* (MIT Press, 1989) and *Corporate Finance: A Valuation Approach* (with Oded Sarig, McGraw-Hill, 1997). Benninga is the editor of the *European Finance Review*.

Simon Benninga
Wharton School, University of Pennsylvania
benninga@wharton.upenn.edu
http://finance.wharton.upenn.edu/~benninga

Zvi Wiener is an assistant professor in the Finance Department of the Business School at the Hebrew University in Jerusalem, Israel. He is currently visiting professor of finance at the Olin School of Business at Washington University, St. Louis. After receiving his Ph.D. in mathematics from the Weizmann Institute, he was a postdoctoral fellow at the Wharton School of the University of Pennsylvania and subsequently worked in the financial derivatives research group at Lehman Brothers. His finance research concentrates on the pricing of derivative securities, value at risk, computational finance, and stochastic dominance.

Zvi Wiener
Finance Department, Business School
Hebrew University, Jerusalem, Israel
msweiner@pluto.mscc.huji.ac.il
http://pluto.mscc.huji.ac.il/~mswiener/zvi.html

**ELECTRONIC SUBSCRIPTIONS**

Included in the distribution for each electronic subscription is the file `BinomialOption.nb` containing *Mathematica* code for the material described in this article.