HW4: Continuous models 2

Marc Luque, NIU: 1567604
Sergi Cucala, NIU: 1570044
Carlo Sala, NIU: 1570775
Júlia Albero, NIU:1566550

Financial Engineering

Exercise 1

We will program a binomial tree algorithm to compute option prices.

The required variables to build up a binomial tree are the stock price S, the upward allowed movement u and the downward allowed movement v. Remember that $0 < v < 1 < u$. We also need the number of time steps to be used N.

We choose to store the tree on a lower triangular matrix T where rows represents depth of the tree, starting at 0, where the initial stock price S lies, this is, $T[0,0] = S$, and going down up to depth N. Therefore, we need a matrix with $N+1$ rows.

Given a particular depth $0 \leq i \leq N$, columns will represent the nodes (scenarios for the stock price). Now, we know from the theory that, given such i, we have $i+1$ nodes: we must end up with stock price at either $v^i S$ (if the stock price had decreased at every time step), $uv^{i-1}S$ (same but price increased once instead), $u^2v^{i-2}S$, ..., $u^{i-1}vS$ or u^iS. Notice that each node can be written as $T[i,j] = u^j v^{i-j} S$ for a certain $0 \leq j \leq i \leq N$. In particular, if we want to stop at the $N-$th time step, at that time step we have $N+1$ nodes and, therefore, our matrix needs $N+1$ columns as well.

Lastly, we have shown that we can write $T[i,j] = u^j v^{i-j} S$ for $0 \leq j \leq i \leq N$, but, in R language, both rows and columns are counted starting at 1 instead of 0, so we need to reorganize indices. We end up with a tree whose entries are

$$T[i,j] = u^{j-1} v^{(i-1)-(j-1)} S = u^{j-1} v^{i-j} S \text{ for } 1 \leq j \leq i \leq N+1 \tag{1}$$

Lets now compute the risk neutral probability of an upward moment q. For this we need the values u, v and the interest rates r with the time step dt (recall that the probability of a downward movement will be $(1-q)$). Recall from the theory, that by a no-arbitrage argument we have:

$$S(1 + rdt) = quS + (1-q)vS.$$

Assuming $S \neq 0$, we can solve the previous equation for q (also keep in mind that $v < 1 < u$, so $u - v \neq 0$):

$$S(1+rdt) = quS + (1-q)vS \Leftrightarrow 1 + rdt = qu + (1-q)v \Leftrightarrow 1 + rdt = q(u-v) + v$$
$$\Leftrightarrow 1 - v + rdt = q(u-v) \Leftrightarrow q = \frac{1 - v + rdt}{u - v}$$

Lets now write a function that computes the value of an option recursively. The first task of the option is to create an empty tree which we will fill with the option value recursively. An $N+1 \times N+1$ zero matrix P will do it, as we need this matrix to store the option values at each node of the tree, so it must have the same dimensions as T.

Then we fill the last nodes of the tree with the payoff function (without premiums because that's exactly what we want to compute at the end!) (our function should take a call or a put option) so

$$P[N+1,j] = payoff(K, T[N+1,j]) \; \forall 1 \leq j \leq N+1 \tag{2}$$

where $payoff(K,S)$ is the option with strike K payoff, when the final stock price is S. We already programmed the payoff function in HW3, so we are just copy-pasting it from there.

Now we use a loop to fill in the nodes one step backward using the formula:

$$V(t) = \frac{V(t+dt)^+ q + V(r + dt^-(1-q)}{1 + rdt}$$

where $V(t)$ is the value of the option at t, $V(t+dt)^+$ means the value of the option at $t+dt$ in the upward node, and similarly for $V(t+dt)^-$. Note that the option price we are looking for is $V(0)$.

For our matrix P, the previous formula translates into

$$P[i,j] = \frac{P[i+1,j+1]q + P[i+1,j](1-q)}{1+rdt}, \; 1 \leq j \leq i \leq N \tag{3}$$

and the option price is $P[1,1]$.

Summarizing, in order to obtain the price of an option, we build a binomial tree T with coefficients as (1) and fill a tree P with the option values at each node of T computed recursively using formulas (2) and (3). Finally, take $P[1,1]$ as the option price.

Exercise 2

As an application of Exercise 1, we are computing the price of a CALL option with strike ATM (At The Money, i.e. the current underlying trading price), therefore we take $K = S$, with underlying currently trading at USD 100, so $S = 100$, and maturing in one year.

We are using a monthly time step and an annualized interest rate of 10%, this is, $r = 0.1$, $dt = 1/12$ as we use a monthly time step and here time has to be expressed in years and we have $N = 12$ time steps in a year, which is the option maturity.

The stock is allowed to move upwards or downwards by 1% each month, meaning $u = 1.01$ and $v = 0.99$.

Putting all this in the function from the previous exercise outputs

```
> binomial_option(type="call",u=1.01,v=0.99,dt=1/12,r=0.1,K=100,S=100,N=12)
[1] 9.478797
```

so the price of such CALL option is $9.48.