PSTAT 276 HW1

Emily Lu

April 27, 2020

4. Consider the up-and-in barrier Call: it pays $V_N = (S_N - K)_+$ but only if $\max_n S_n > B$, i.e. at some point between 0 and T, the stock price exceeded the level B. If this happens, we say that the option got "knocked in". The barrier option is path-dependent. Take a binomial tree with $u = 1.05$, $d = 0.95$, $r = 0.01$ and $N = 8$. The barrier parameters are $S_0 = 10$, $K = 11$ and B to be specified below. There are a total of $2^N = 256$ scenarios in this tree. By directly enumerating them as $\omega^{(1)} = HHHHHHHH$, $\omega^{(2)} = HHHHHHHT$, ..., $\omega^{(256)} = TTTTTTTT$ and computing $\mathbb{Q}(\omega^{(i)})$, find the price of the up-and-in barrier Call:

$$\mathbb{E}^Q\left[\frac{V_N}{(1+r)^N}\right] = \frac{1}{(1+r)^N}\sum_{i=1}^{256}\mathbb{Q}(\omega^{(i)})V_N(\omega^{(i)})$$

The provided R code in class shows how to enumerate the scenarios by translating between the index i of $\omega^{(i)}$ and the individual coin tosses $\omega_k^{(i)}$. As an example, it computes the final stock price $S_N(\omega_1 ... \omega_N)$. You will need to modify the code to compute the barrier Call payoffs.

For each $B = 11, 11.5, 12, 12.5, 13$ report: (i) number of scenarios that have nonzero payoff; (ii) risk-neutral probability of being knocked-in, (iii) no-arbitrage price of the barrier Call. Note: if $B = 11 = K$ then the barrier feature doesn't do anything and the answer you get is the same as for ordinary Call.

```r
u = 1.05; d = 0.95; r = 0.01; N = 8; S0 = 10; M = 256
q <- (1+r-d)/(u-d)
p <- 1-q

genPath <- function(M, r=r, n=N, u=u, d=d, S0=S0){
  q <- (1+r-d)/(u-d)
  S <- array(0,dim=c(M,n+1))
  Q <- array(0, dim=c(M,n+1))
  S[,1] <- S0
  for (i in 1:n) {
    UUi <- runif(M)
    S[,i+1] <- S[,i]*u^(UUi < q)*d^(UUi >= q)
    }
  return (list(S=S, UUi=UUi))
}
paths <- genPath(256, r=r, n=N, u=u, d=d, S0=S0)
SN <- paths$S
UUi <- paths$UUi

for (i in 1:M){
  if (UUi[i] < q)
    UUi[i] = 1
  else
    UUi[i] = 0
}

knock_in <- function(M, B, K, S=SN,r=r, n=N, u=u, d=d, S0=S0, w = UUi){
  Q <- 0
```

```r
  Payoff <- rep(0, M)
  nonzero <- 0
  for (j in 1: M){
    Q <- Q + q^sum(w)*(1-q)^(2^8-sum(w))
    if (max(S[j, 1:N+1]) > B)
      Payoff[j] <- pmax(S[j, N+1]-K, 0)
    if (Payoff[j] > 0)
      nonzero <- nonzero + 1
  }
  Price <- sum(Q*Payoff)/(1+r)^N
  sprintf('Nonzero count is %s.', nonzero)
  sprintf('Risk prob. is %s.', Q)
  sprintf('Price is %s.', Price)
  return(0)
}
knock_in(256, 11, 11, S=SN,r=r, n=N, u=u, d=d, S0=S0, w = UUi)
```

```
## [1] 0
```

6. Consider a 4-period model for the EUR/USD exchange rate with S0 = 1.05, u = 1.01, d = 0.99 and r = 0 (no interest rates!).

(a) Price a "Hit Box Option" that extends from t = 1.9 to t = 4.1 and from St = 1.03 to St = 1.07 (i.e. you need to consider S1, S2, S3, S4 to determine the payoff);

```r
HitBox <- function(S, tl, tr, Bd, Bu){
  index_l <- max(1, ceiling(tl))
  index_r <- min(length(S), floor(tr))
  t = index_l
  while(t <= index_r){
    if (S[t] <= Bu && S[t] >= Bd){
      return(TRUE)
    } else {
      t = t +1
    }
  }
  return(FALSE)
}



BoxOption <- function(hit, tl, tr, Bd, Bu, N,
u=1.01, d=0.99, S0=1.05, r=0){
# Compute all possible scenarios and probabilities
  q <- (1+r-d)/(u-d)
  S <- rep(0, 2^N)
  w <- expand.grid(rep(list(0:1), N))
  for (i in 1:2^N) {
    numH <- sum(w[i,] == 1)
    numT <- sum(w[i,] == 0)
    S[i] <- S0*u^numH*d^numT
  }

  Payoff <- rep(0, 2^N)
  Q <- 0
```

```r
  for (j in 1: (2^N)){
    Payoff[j] <- ifelse(HitBox(S[j], tl, tr, Bd, Bu),
          ifelse(hit, 1, 0), ifelse(hit, 0, 1))
    Q <- Q + q^sum(w[j, ])*(1-q)^(8-sum(w[j, ]))
    }
    Price <- sum(Q*Payoff)/(1+r)^N
    return(Price)
}
# BoxOption(1, 1.9, 4.1, 1.03, 1.07, 4)
# BoxOption(0, 1.9, 4.1, 1.045, 1.055, 4)
```

(b) Price a "Miss Box Option" that extends from t = 1.9 to t = 4.1 and from St = 1.045 to St = 1.055. Bonus (+3pt): write code (based on the R code also used in Q4) to automate the pricing of this option for any user-specified Box parameters and any number of periods N.

```r
# BoxOption(1, 1.9, 4.1, 1.03, 1.07, 4)
```