Module 5.3: Arithmetic Brownian Motion-Based Binomial Models
R Commentary

See module *Ch 5.3 Valuation ABM Binomial OVM*. There are two main test programs, *Valuation ABM Binomial OVM European-Style Test.R* and *Valuation ABM Binomial OVM American-Style Test.R*. Technical functions are contained in *ESABMBINOVM Backward Recursion Function.R* and *ASABMBINOVM Backward Recursion Function.R*.

Valuation ABM Binomial OVM European-Style Test.R

This program examines a particular set of data for one option. We specifically examine the ABM case that calibrates to the GBM case of 30% volatility.

```
source('ESABMBINOVM Backward Recursion Function.R')
# Test inputs
inputStockPrice = 100.0          # Need "input" as using variable names below
inputStrikePrice = 100.0         # In currency units, numeric
inputInterestRate = 5.0          # In percent
inputDividendYield = 0.0         # In percent
inputVolatility = 29.88476829    # 29.88476829   # In dollars, annualized
inputTimeToMaturity = 1.0        # In fraction of year
inputType = 1L                   # 1 for call, -1 for put
inputNumberOfSteps = as.integer(500)   # Or use L: 1000L
inputPayoutType = 1L             # 1 Plain vanilla, 2 digital
inputEMMProbability = 50.0       # In percent
inputDigitalPayout = 100.0
```

The analysis is provided in *ESABMBINOVM Backward Recursion Function.R*. The key to understanding this code is the outer loop runs backward and the option values are stored in the same vector. As it loops backward, one value in the vector is no longer referenced. The final answer is in the first element of the vector.

```
    for (TimeStep in NumberOfSteps:0){
      for (StateStep in 0:TimeStep){
        if(TimeStep == NumberOfSteps){
          S <- StockPrice + StateStep*Up + (TimeStep - StateStep)*Down
          CallValue[StateStep+1] <- max(0, S - StrikePrice)
          PutValue[StateStep+1] <- max(0, StrikePrice - S)
          if(S > StrikePrice){
            DigitalCallValue[StateStep+1] <- DigitalPayout
            DigitalPutValue[StateStep+1] <- 0
          } else {
            DigitalCallValue[StateStep+1] <- 0
            DigitalPutValue[StateStep+1] <- DigitalPayout
          }
        } else {
          S <- StockPrice + StateStep*Up + (TimeStep - StateStep)*Down
          phi <- (S*R - Down)/(Up - Down)
          CallValue[StateStep+1] <- (1/PeriodRate) *
            (phi*CallValue[StateStep+2] + (1 - phi)*CallValue[StateStep+1])
          PutValue[StateStep+1] <- (1/PeriodRate) *
            (phi*PutValue[StateStep+2] + (1 - phi)*PutValue[StateStep+1])
          DigitalCallValue[StateStep+1] <- (1/PeriodRate) *
            (phi*DigitalCallValue[StateStep+2] +
            (1 - phi)*DigitalCallValue[StateStep+1])
          DigitalPutValue[StateStep+1] <- (1/PeriodRate) *
            (phi*DigitalPutValue[StateStep+2] +
            (1 - phi)*DigitalPutValue[StateStep+1])
        }
      }
    }
```

Valuation ABM Binomial OVM American-Style Test.R

Again this program examines a particular set of data for one option. The analysis is provided in *ASABMBINOVM Backward Recursion Function.R*. The code is nearly identical to the European-style

calculation except additional values are considered at each node. Warning: This program takes a while to run.