Module 5.5: Arithmetic Brownian Motion-Based Option Valuation Models
R Commentary

See *Ch 5.5 Valuation ABM CT OVM*. The test program is in *ABMOVM Test.R*. The functions are contained in ABMOVM Functions.R.

ABMOVM Test.R (Selected Excerpts and Output)

The ABMOVM test program follows GBMOVM closely.

```
# 5.5 ABMOVM Test.R
# Arithmetic Brownian Motion Option Valuation Model
# rmarkdown::render("5.5 ABMOVM Test.R", "word_document")
rm(list = ls())  # Take out the Environment "trash"
cat("\014") # Clear Console, making error checking easier.
while (!is.null(dev.list()))dev.off() # Clear old plots
par(family = 'Times New Roman') # Globally set fonts for graphs
# Generic test inputs
inputStockPrice = 100.0
inputStrikePrice = 100.0
inputInterestRate = 5.0        # In percent
inputDividendYield = 0.0       # In percent
inputVolatility = 29.88 #476829  # In dollars, annualized
inputTimeToMaturity = 1
inputType = 1 # 1 for call, -1 for put
NumberOfObservations = 200
LowerBound = 0.5*inputStockPrice # Note centering on original values
UpperBound = 1.5*inputStockPrice
LowerBoundNL = 0.9*inputStockPrice # Note centering on original values
UpperBoundNL = 1.1*inputStockPrice
# Plot footers
TS = paste0('S=', inputStockPrice)
TX = paste0(',X=', inputStrikePrice)
TR = paste0(',r=', inputInterestRate)
Td = paste0(',d=', inputDividendYield, ',')
TV = paste0('Vol=', inputVolatility)
TT = paste0(',T=',inputTimeToMaturity)
sTitle = paste0(TS, TX, TR, Td, TV, TT)
#
# Available functions
#
#  ABMInputData - list of inputs with associated names
#  PV1(Maturity, Rate) - present value of $1
#  B = ABMInputData
#  d1(B) - value of d1
#  d2(B) - value of d2
#  n(d) - standard normal PDF, given scalar d
#  N(d) - standard normal CDF, given scalar d
#  ABMOptionValue(B) - option value, type = 1 is call, type = -1 is put
#  OptionLowerBound(B) - option lower bounds
#  OptionUpperBound(B) - option upper bounds
#
#  Input matrix
#  ABMInputData - list of inputs with associated names
ABMInputData <- list(inputStockPrice, inputStrikePrice, inputInterestRate,
  inputDividendYield, inputVolatility, inputTimeToMaturity, inputType)
names(ABMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
  "DividendYield", "Volatility", "TimeToMaturity", "Type")
source("ABMOVM Functions.R")
#
# Test functions
#
# Basic functions
UIValue = ABMInputData$StockPrice
TestPV1 = PV1(inputTimeToMaturity, inputInterestRate)
```

```
Testdn = dn(ABMInputData)
Testn = n(dn(ABMInputData))
TestN = N(dn(ABMInputData))
Testdn; Testn; TestN
# Boundaries
ABMInputData$Type = 1
CallLowerBound = OptionLowerBound(ABMInputData)
CallUpperBound = OptionUpperBound(ABMInputData)
CallValue = ABMOptionValue(ABMInputData)
CallLowerBound; CallUpperBound; CallValue
ABMInputData$Type = -1
PutLowerBound = OptionLowerBound(ABMInputData)
PutUpperBound = OptionUpperBound(ABMInputData)
PutValue = ABMOptionValue(ABMInputData)
PutLowerBound; PutUpperBound; PutValue
…
#
# Illustrations with plots
#
StepSize = (UpperBound - LowerBound)/NumberOfObservations
StockPrice <- c(1:NumberOfObservations)
CallValue <- c(1:NumberOfObservations)
PutValue <- c(1:NumberOfObservations)
CallLowerBound <- c(1:NumberOfObservations)
PutLowerBound <- c(1:NumberOfObservations)
CallUpperBound <- c(1:NumberOfObservations)
PutUpperBound <- c(1:NumberOfObservations)
CallTimeValue <- c(1:NumberOfObservations)
PutTimeValue <- c(1:NumberOfObservations)
for(i in 1:NumberOfObservations){
  StockPrice[i] <- as.double(LowerBound + (i-1)*StepSize)
  ABMInputData$StockPrice = StockPrice[i]
  ABMInputData$Type = 1
  CallLowerBound[i] <- OptionLowerBound(ABMInputData)
  CallUpperBound[i] <- OptionUpperBound(ABMInputData)
  CallValue[i] <- ABMOptionValue(ABMInputData)
  CallTimeValue[i] <- CallValue[i] - CallLowerBound[i]
  ABMInputData$Type = -1
  PutLowerBound[i] <- OptionLowerBound(ABMInputData)
  PutUpperBound[i] <- OptionUpperBound(ABMInputData)
  PutValue[i] <- ABMOptionValue(ABMInputData)
  PutTimeValue[i] <- PutValue[i] - PutLowerBound[i]
}
ABMInputData$StockPrice = inputStockPrice
# Option value plots
MaxValue = max(CallValue, CallLowerBound)
MinValue = min(CallValue, CallLowerBound)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Call Value","Lower Bound")
mTitle = "ABM Call Value"
xTitle = "Stock Price"
yTitle = "Call Value"
lTitle = "Parameter"
plot(StockPrice, CallValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
lines(StockPrice, CallLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 2, cex = 0.5)
legend("topleft", legtxt, cex = 0.75, lwd = c(1, 1), lty = c(1, 1),
  col = c("black","black"), pch = c(1, 2), bty = "n", title = lTitle)
# Put value
MaxValue = max(PutValue, PutLowerBound)
```

```
MinValue = min(PutValue, PutLowerBound)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Put Value","Lower Bound")
mTitle = "ABM Put Value"
xTitle = "Stock Price"
yTitle = "Put Value"
lTitle = "Parameter"
plot(StockPrice, PutValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
lines(StockPrice, PutLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 2, cex = 0.5)
legend("topright", legtxt, cex = 0.75, lwd = c(1, 1), lty = c(1, 1),
  col = c("black","black"), pch = c(1, 2), bty = "n", title = lTitle)
# All together now
MaxValue = max(CallValue, CallLowerBound, PutValue, PutLowerBound)
MinValue = min(CallValue, CallLowerBound, PutValue, PutLowerBound)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Call Value", "Call Lower Bound", "Put Value", "Put Lower Bound")
mTitle = "ABM Option Values"
xTitle = "Stock Price"
yTitle = "Option Value"
lTitle = "Parameter"
plot(StockPrice, CallValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
lines(StockPrice, CallLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 2, cex = 0.5)
lines(StockPrice, PutValue, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 3, cex = 0.5)
lines(StockPrice, PutLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 4, cex = 0.5)
legend("top", legtxt, cex = 0.75, lwd = c(1, 1, 1, 1), lty = c(1, 1, 1, 1),
  col = c("black","black","black","black"), pch = c(1, 2, 3, 4),
  bty = "n", title = lTitle)
#
# Option time value plots
#
MaxValue = max(CallTimeValue); MinValue = min(CallTimeValue)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Call Value","Lower Bound")
mTitle = "ABM Call Time Value"
xTitle = "Stock Price"
yTitle = "Call Time Value"
lTitle = "Parameter"
plot(StockPrice, CallTimeValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
# Put time value
MaxValue = max(PutTimeValue); MinValue = min(PutTimeValue)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Put Value","Lower Bound")
mTitle = "ABM Put Time Value"
xTitle = "Stock Price"
yTitle = "Put Time Value"
lTitle = "Parameter"
```

```
plot(StockPrice, PutTimeValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
#
# Option boundary plots
#
MaxValue = max(CallValue, CallLowerBound, CallUpperBound)
MinValue = min(CallValue, CallLowerBound, CallUpperBound)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Call Value", "Call Lower Bound", "Call Upper Bound")
mTitle = "ABM Option Values"
xTitle = "Stock Price"
yTitle = "Value"
lTitle = "Parameter"
plot(StockPrice, CallValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
lines(StockPrice, CallLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 2, cex = 0.5)
lines(StockPrice, CallUpperBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 3, cex = 0.5)
legend("topleft", legtxt, cex = 0.75, lwd = c(1, 1, 1), lty = c(1, 1, 1),
  col = c("black","black","black"), pch = c(1, 2, 3), bty = "n", title = lTitle)
# Put bounds
MaxValue = max(PutValue, PutLowerBound, PutUpperBound)
MinValue = min(PutValue, PutLowerBound, PutUpperBound)
ylim1 = c(1:2); ylim1[1] = MinValue; ylim1[2] = MaxValue
MaxValue = max(StockPrice); MinValue = min(StockPrice)
xlim1 = c(1:2); xlim1[1] = MinValue; xlim1[2] = MaxValue
legtxt = c("Put Value", "Put Lower Bound", "Put Upper Bound")
mTitle = "ABM Option Values"
xTitle = "Stock Price"
yTitle = "Value"
lTitle = "Parameter"
plot(StockPrice, PutValue, type = "b", main = mTitle,
  sub = sTitle, xlab = xTitle, ylab = yTitle, col = "black", xlim = xlim1,
  ylim = ylim1, pch = 1, cex = 0.5)
lines(StockPrice, PutLowerBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 2, cex = 0.5)
lines(StockPrice, PutUpperBound, type = "b", col ="black", xlim = xlim1,
  ylim = ylim1, pch = 3, cex = 0.5)
legend("bottomleft", legtxt, cex = 0.75, lwd = c(1, 1, 1), lty = c(1, 1, 1),
  col = c("black","black","black"), pch = c(1, 2, 3), bty = "n", title = lTitle)
```

ABM Functions.R

Again, ABM Functions.R follows GBMFunctions.R closely.

```
# ABMOVM Functions.R
#
#  INPUT STRUCTURE
#  ABMInputData - list of inputs with associated names; percent, not decimal
#  ABMInputData <- list(inputStockPrice, inputStrikePrice, inputInterestRate,
#    inputDividendYield, inputVolatility, inputTimeToMaturity, inputType)
#  names(ABMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
#    "DividendYield", "Volatility", "TimeToMaturity", "Type")
#
# Available functions
#  PV1(Maturity, Rate) - present value of $1
#  B = ABMInputData
#  d1(B) - value of d1
#  d2(B) - value of d2
#  n(d) - standard normal PDF, given scalar d
#  N(d) - standard normal CDF, given scalar d
```

```
#  ABMOptionValue(B) - option value, type = 1 is call, type = -1 is put
#  OptionLowerBound(B) - option lower bounds
#  OptionUpperBound(B) - option upper bounds
#
# Functions for ABMOVM-related calculations
#
PV1 = function(Maturity, Rate){
  return(exp( -(Rate/100.0) * Maturity) )
}
```

The key function in ABMOVM Functions.R is the valuation function that is written generically to be able to value either the call or put option. Within the ABMOVM, it is convenient to have a separate function to compute the adjusted volatility term.

```
# Adjusted volatility to use in dn and option values
AdjustedSigma = function(B){
  with(B, {
    if (abs(InterestRate - DividendYield)<0.00001){
      AdjSigma <- Volatility * sqrt(TimeToMaturity)
    } else {
      AdjSigma <- Volatility *
        sqrt( (( PV1(TimeToMaturity,-2*(InterestRate-DividendYield)) )-1) /
                (2*(InterestRate-DividendYield)/100) )
    }
    return(AdjSigma)
  })
}
# dn - functions used in ABMOVM
# with: Evaluates R expressions based on a set of data (B here)
dn = function(B){
  with(B, {
    AdjSigma <- AdjustedSigma(B)
    Num = StockPrice*PV1(TimeToMaturity, -(InterestRate-DividendYield)) -
      StrikePrice
    Den = AdjSigma
    return( Num/Den )
  })
}
# Normal distribution functions
# n - probability density function of standard normal (0,1)
n = function(d){
  return( exp( -(d^2) / 2.0 ) / ( sqrt( 2.0 * pi ) ) )
}
# N - cumulative distribution function of standard normal (0,1)
N = function(d){
  return( as.numeric(integrate(n,-Inf,d)[1]) )
}
```

The generic ABMOVM valuation function is relatively straightforward.

```
# ABMOVM
ABMOptionValue = function(B){
  with(B, {
    AdjSigma <- AdjustedSigma(B)
    OptionValue = Type * ( StockPrice * PV1(TimeToMaturity, DividendYield) -
      StrikePrice * PV1(TimeToMaturity, InterestRate) ) * N(Type * dn(B)) +
      PV1(TimeToMaturity, InterestRate) * AdjSigma * n(dn(B))
    LowerBound = Type * StockPrice * PV1(TimeToMaturity, DividendYield) -
      Type * StrikePrice * PV1(TimeToMaturity, InterestRate)
    return( max(OptionValue, LowerBound) )
  })
}
# Lower bound
OptionLowerBound = function(B){
  with(B, {
    LowerBound = Type * StockPrice * PV1(TimeToMaturity, DividendYield) -
      Type * StrikePrice * PV1(TimeToMaturity, InterestRate)
```

```
    LowerBound = max(0, LowerBound)
    return( LowerBound )
  })
}
# Upper bound
OptionUpperBound = function(B){
  with(B, {
    if(Type == 1)UpperBound = StockPrice * PV1(TimeToMaturity, DividendYield)
    if(Type == -1)UpperBound = StrikePrice * PV1(TimeToMaturity, InterestRate)
    return( UpperBound )
  })
}
```

With these user-defined functions built in R, several plots are produced. Again, most of the R code is similar to GBMOVM.