Module 8.3: SRM GBM-Based Option Models
R Commentary

See *SRM GBM CT OVM.R.*

SRM GBM CT OVM Test.R

```
# SRM GBM CT OVM Test.R
# GBMOVM, ES and AS BOVMs
# rmarkdown::render("SRM GBM CT OVM 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
# Libraries
#  beepr - functions for beeping to let you know when program is finished
Packages <- c("beepr")
if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(Packages, rownames(installed.packages())))
} # Make sure libraries are installed on this computer
lapply(Packages, library, character.only=TRUE) # Load and attach libraries
rm(Packages)
# Functions defined in:
#   GBMOVM w Greeks Functions.R
#   ESGBMBINOVM With SRM Functions.R (European-style only)
#   ASGBMBINOVM With SRM Functions.R (American-style also)
#
# Test inputs
# GBMOVM inputs
inputStockPrice = 100       # Need "input" as using variable names below
inputStrikePrice = 100
inputInterestRate = 5.0        # In percent
inputDividendYield = 5.0       # In percent
inputVolatility = 30.0         # In percent
inputTimeToMaturity = 1.0
inputType = 1                  # 1 for call, -1 for put
# Additional BOVM inputs
inputEMMProbability = 50.0     # In percent
inputGreekIncrement = 1.0      # In percent of greek underlying variable
inputNumberOfSteps = as.integer(250) # Or use L: 1000L
inputPayoutType = 1L        # 1 Plain vanilla, 2 digital (digital not built)
inputStyle = 2              # 1 for terminal ES, 2 for AS
inputDigitalPayout = 100    # In dollars
# Plot ranges
LowerBound = inputStockPrice*0.5 # Stock price
UpperBound = inputStockPrice*1.5
LowerBoundV = inputVolatility*0.5 # Volatility
UpperBoundV = inputVolatility*1.5
LowerBoundT = inputTimeToMaturity*0.5 # Time to maturity
UpperBoundT = inputTimeToMaturity*1.5
LowerBoundD = 0.0 # Dividend yield
UpperBoundD = 25.0

NumberOfObservations = 51
#
#  BINInputData - list of inputs with associated names
#
BINInputData <- list(inputStockPrice, inputStrikePrice, inputInterestRate,
  inputDividendYield, inputVolatility, inputTimeToMaturity, inputType,
  inputNumberOfSteps, inputPayoutType, inputStyle,
  inputEMMProbability, inputDigitalPayout, inputGreekIncrement)
names(BINInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
  "DividendYield", "Volatility", "TimeToMaturity", "Type",
```

```
  "NumberOfSteps", "PayoutType", "Style",
  "EMMProbability", "DigitalPayout", "GreekIncrement")
#
# GBM list
#
GBMInputData <- list(inputStockPrice, inputStrikePrice, inputInterestRate,
  inputDividendYield, inputVolatility, inputTimeToMaturity, inputType)
names(GBMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
  "DividendYield", "Volatility", "TimeToMaturity", "Type")
#
# Source the appropriate functions
#
source("GBMOVM With SRM Functions.R")
source('ESGBMBINOVM With SRM Functions.R')
source('ASGBMBINOVM With SRM Functions.R')
# Console Rounding Setting
RDigits <- 4
# source("Core Functions Tests.R")
source("GBMOVM Stock Price Plots.R")
source("GBMOVM Volatility Plots.R")
source("GBMOVM Time To Maturity Plots.R")
source("GBMOVM Dividend Yield Plots.R")
beep(sound = 3, print('Finished')) # fanfare
```

ESGBMBINOVM With SRM Functions.R (Selected Excerpts and Output)

The complete function for delta is given based on European-style options.

```
#
# Delta Direct Enhanced Method -- have to do full backward recursion
#
ESBINOptionDeltaDirectEnh <- function(B){
  with(B, {
    OriginalTimeToMaturity <- TimeToMaturity
    OriginalStockPrice <- StockPrice
    OriginalNumberOfSteps <- NumberOfSteps
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    TimeToMaturity <- TimeToMaturity + 2.0*Delta
    NumberOfSteps <- NumberOfSteps + 2
    OHigh <- OLow <- SHigh <- SLow <- 0
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Rate <- InterestRate/100.0  # Local variable, in decimal
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)             # Periodic rate (1+r)
    PeriodDriftRate <- exp(DriftRate*Delta)   # Periodic rate (1+r)
    AdjustedStockPrice <- StockPrice
    Prob <- EMMProbability/100.0
    A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    Down <- PeriodDriftRate / (Prob*A + (1-Prob))
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    TNS <- NumberOfSteps + 1
    OptionValue <- c(1:TNS)
    # OptionValue <- c(1:NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
# Inner loop
      for (i in 0:TimeStep){
        if(TimeStep == NumberOfSteps){  # At expiration
          if(Type == 1){    # Plain vanilla call
            Moneyness <- (Up^i) * (Down^(TimeStep-i)) * AdjustedStockPrice -
```

```
              StrikePrice
            OptionValue[i+1] <- max(0, Moneyness)
          }
          if(Type == -1){   # Plain vanilla put
            Moneyness <- StrikePrice -
              ((Up^i)*(Down^(TimeStep-i)) * AdjustedStockPrice)
            OptionValue[i+1] <- max(0, Moneyness)
          }
        } else { # Prior to expiration
          OptionValue[i+1] <- (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
        }
        if(TimeStep == 2){
          if(i == 0){
            OLow <- OptionValue[i+1]
            SLow <- (Up^i) * (Down^(TimeStep-i)) * AdjustedStockPrice
            B$StockPrice <- SLow
            B$TimeToMaturity <- OriginalTimeToMaturity
# Check lower boundary conditions
            if(PayoutType == 1){    # Plain vanilla option
              LowerBoundL <- ESBINOptionLowerBound(B)
            } else {
              LowerBoundL <- 0
            }
            LowerBoundL <- max(0, LowerBoundL)
            OLow <- max(OLow, LowerBoundL)
            B$TimeToMaturity <- OriginalTimeToMaturity + 2*Delta
            B$StockPrice <- OriginalStockPrice
          }
          if(i == 2){
            OHigh <- OptionValue[i+1]
            SHigh <- (Up^i) * (Down^(TimeStep-i)) * AdjustedStockPrice
            B$StockPrice <- SHigh
            B$TimeToMaturity <- OriginalTimeToMaturity - 2*Delta
# Check lower boundary conditions
            if(PayoutType == 1){    # Plain vanilla option
              LowerBoundH <- ESBINOptionLowerBound(B)
            } else {
              LowerBoundH <- 0
            }
            LowerBoundH <- max(0, LowerBoundH)
            OHigh <- max(OHigh, LowerBoundH)
            B$TimeToMaturity <- OriginalTimeToMaturity + 2*Delta
            B$StockPrice <- OriginalStockPrice
          }
        }
      }
    }
    Value <- (OHigh - OLow) / (SHigh - SLow)
    return(Value)
  })
}
```

ASGBMBINOVM With SRM Functions.R

```
# ASGBMBINOVM With SRM Functions.R
#  American-style, geometric Brownian motion, binomial OVM
# Present value function (See ESGBMBINOVM With SRM Functions.R)
# Binomial probability
source('ESGBMBINOVM With SRM Functions.R')
# Built on ASGBMBINOVM Functions.R
source('ASGBMBINOVM Functions.R')
#
# Delta Direct -- have to do full backward recursion
```

```
#
ASBINOptionDeltaDirect <- function(B){
  with(B, {
    # OriginalTimeToMaturity <- TimeToMaturity
    # OriginalStockPrice <- StockPrice
    OHigh <- OLow <- SHigh <- SLow <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    Prob <- EMMProbability/100.0
    A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    Down <- PeriodDriftRate / (Prob*A + (1-Prob))
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){ # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else {                           # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - dTimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
        if(TimeStep == 1){ # Increment time to maturity and time to PV Div
          if(i == 0){
            OLow <- OptionValue[i+1]
            SLow <- S
          }
          if(i == 1){
            OHigh <- OptionValue[i+1]
            SHigh <- S
          }
        }
      }
    }
    Value <- (OHigh - OLow) / (SHigh - SLow)
    return(Value)
  })
}
#
# Delta Direct Enhanced (+2 Delta) -- have to do full backward recursion
#
ASBINOptionDeltaDirectEnh <- function(B){
  with(B, {
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    TimeToMaturity <- TimeToMaturity + 2.0*Delta
    NumberOfSteps <- NumberOfSteps + 2
    OHigh <- OLow <- SHigh <- SLow <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
```

```
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    AdjustedStockPrice <- StockPrice
    Prob <- EMMProbability/100.0
    A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    Down <- PeriodDriftRate / (Prob*A + (1-Prob))
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){ # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else { # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - dTimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
        if(TimeStep == 2){
          if(i == 0){
            OLow <- OptionValue[i+1]
            SLow <- S
          }
          if(i == 2){
            OHigh <- OptionValue[i+1]
            SHigh <- S
          }
        }
      }
    }
    Value <- (OHigh - OLow) / (SHigh - SLow)
    return(Value)
  })
}
#
# Gamma Direct
#
ASBINOptionGammaDirect <- function(B){
  with(B, {
    OHigh <- OLow <- SHigh <- SLow <- SMid <- OMid <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    AdjustedStockPrice <- StockPrice
    Prob <- EMMProbability/100.0
```

```
    A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    Down <- PeriodDriftRate / (Prob*A + (1-Prob))
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){ # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else { # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - dTimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
        if(TimeStep == 2){
          if(i == 0){
            OLow <- OptionValue[i+1]
            SLow <- S
          }
          if(i == 1){
            OMid <- OptionValue[i+1]
            SMid <- S
          }
          if(i == 2){
            OHigh <- OptionValue[i+1]
            SHigh <- S
          }
        }
      }
    }
    Value <- ( (OHigh - OMid)/(SHigh - SMid) -
      (OMid - OLow)/(SMid - SLow) ) / (0.5*(SHigh - SLow))
    return(Value)
  })
}
#
# Gamma Direct Enhanced
#
ASBINOptionGammaDirectEnh <- function(B){
  with(B, {
    OriginalTimeToMaturity <- TimeToMaturity
    OriginalStockPrice <- StockPrice
    OriginalNumberOfSteps <- NumberOfSteps
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    TimeToMaturity <- TimeToMaturity + 2.0*Delta
    NumberOfSteps <- NumberOfSteps + 2
    OHigh <- OLow <- SHigh <- SLow <- SMid <- OMid <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    AdjustedStockPrice <- StockPrice
```

```
    Prob <- EMMProbability/100.0
    A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    Down <- PeriodDriftRate / (Prob*A + (1-Prob))
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){ # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else { # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - dTimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
        if(TimeStep == 2){
          if(i == 0){
            OLow <- OptionValue[i+1]
            SLow <- S
          }
          if(i == 1){
            OMid <- OptionValue[i+1]
            SMid <- S
          }
          if(i == 2){
            OHigh <- OptionValue[i+1]
            SHigh <- S
          }
        }
      }
    }
    Value <- ( (OHigh - OMid)/(SHigh - SMid) -
      (OMid - OLow)/(SMid - SLow) ) / (0.5*(SHigh - SLow))
    return(Value)
  })
}
#
# Theta Direct
#
ASBINOptionThetaDirect <- function(B){
  with(B, {
    OriginalTimeToMaturity <- TimeToMaturity
    OriginalStockPrice <- StockPrice
    OHigh <- OLow <- SHigh <- SLow <- SMid <- OMid <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    # Prob <- EMMProbability/100.0
    # A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    # Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    # Down <- PeriodDriftRate / (Prob*A + (1-Prob))
```

```
#
# Thetas corrupted unless S(0) = S(ud): Cox, Ross, Rubinstein
#
    Up <- exp(Sigma*sqrt(Delta))
    Down <- 1/Up
    Prob <- (exp(DriftRate*Delta) - Down) / (Up - Down)
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      dTimeStep <- as.numeric(TimeStep)
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){  # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else { # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - TimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
        if(TimeStep == 2){
          if(i == 1){
            OMid <- OptionValue[i+1]
            SMid <- S
          }
        }
      }
    }
    Value <- (OMid - OptionValue[1])/(2.0*Delta)
    return(Value)
  })
}
#
# Theta Direct Enhanced
#
ASBINOptionThetaDirectEnh <- function(B){
  with(B, {
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    TimeToMaturity <- TimeToMaturity + 2.0*Delta
    NumberOfSteps <- NumberOfSteps + 2
    OHigh <- OLow <- SHigh <- SLow <- SMid <- OMid <- 0
    N <- NumberOfSteps + 1
    OptionValue <- c(1:N)
    Sum <- 0
    Moneyness <- 0
    Value <- 0
    Rate <- InterestRate/100.0
    DriftRate <- (InterestRate - DividendYield)/100.0
    Sigma <- Volatility/100.0   # Local variable, in decimal
    Delta <- TimeToMaturity / as.numeric(NumberOfSteps)
    PeriodRate <- exp(Rate*Delta)
    PeriodDriftRate <- exp(DriftRate*Delta)
    # Prob <- EMMProbability/100.0
    # A <- exp( Sigma*sqrt(Delta)/sqrt(Prob*(1-Prob)) )
    # Up <- ( PeriodDriftRate * A ) / (Prob*A + (1-Prob))
    # Down <- PeriodDriftRate / (Prob*A + (1-Prob))
#
# Thetas corrupted unless S(0) = S(ud): Cox, Ross, Rubinstein
#
    Up <- exp(Sigma*sqrt(Delta))
    Down <- 1/Up
    Prob <- (exp(DriftRate*Delta) - Down) / (Up - Down)
```

```
    if(Up < PeriodRate || Down > PeriodRate)return(-99)
    dSteps <- as.numeric(NumberOfSteps)
    for (TimeStep in NumberOfSteps:0){
      for (i in 0:TimeStep){
        S <- (Up^i) * (Down^(TimeStep-i)) * StockPrice
        if(TimeStep == NumberOfSteps){  # At expiration
          OptionValue[i+1] = max(0, Type*(S - StrikePrice))
        } else { # Prior to expiration
          B$TimeToMaturity <- TimeToMaturity - TimeStep*Delta
          B$StockPrice <- S
          LowerBound <- ASBINOptionLowerBound(B)
          OptionValue[i+1] = (1.0/PeriodRate) * (Prob*OptionValue[i+2] +
            (1.0 - Prob)*OptionValue[i+1])
          OptionValue[i+1] = max(LowerBound, OptionValue[i+1])
        }
      if(TimeStep == 4){
        if(i == 2){
          OMid <- OptionValue[i+1]
          SMid <- S
        }
      }
    }
  }
  Value <- (OMid - OptionValue[1])/(4.0*Delta)
  return(Value)
  })
}
#
# Numerical Greeks
#
# Delta
ASBINOptionDelta <- function(B){
  Original <- B$StockPrice
  Change <- (B$GreekIncrement/100.0)*Original
  High <- Original + Change
  B$StockPrice <- High
  OHigh <- ASBINOptionValue(B)
  Low <- Original - Change
  B$StockPrice <- Low
  OLow <- ASBINOptionValue(B)
  B$StockPrice <- Original
  OptionDelta <- (OHigh - OLow)/(High - Low)
  return( OptionDelta )
}
# Gamma: Goes to zero for large time steps (UNSTABLE)
ASBINOptionGamma <- function(B){
  Original <- B$StockPrice
  Change <- (B$GreekIncrement/100.0)*Original*100
  High <- Original + Change
  B$StockPrice <- High
  OHigh <- ASBINOptionValue(B)
  Low <- Original - Change
  B$StockPrice <- Low
  OLow <- ASBINOptionValue(B)
  B$StockPrice <- Original
  OMid <- ASBINOptionValue(B)
  OptionGamma <- ( (OHigh - OMid) - (OMid - OLow) )/(Change*Change)
  # return(OMid - OLow)
  return(OptionGamma)
}
# Theta
ASBINOptionTheta <- function(B){
  Original <- B$TimeToMaturity
  Change <- (B$GreekIncrement/100.0)*Original*(1/10)
```

```
  High <- Original + Change
  B$TimeToMaturity <- High
  OHigh <- ASBINOptionValue(B)
  Low <- Original - Change
  B$TimeToMaturity <- Low
  OLow <- ASBINOptionValue(B)
  B$TimeToMaturity <- Original
  OptionTheta <- (OHigh - OLow)/(High - Low)
  # Moving time to maturity up is maturity time down (-)
  return( -OptionTheta )
}
# Vega
ASBINOptionVega <- function(B){
  Original <- B$Volatility
  Change <- (B$GreekIncrement/100.0)*Original
  High <- Original + Change
  B$Volatility <- High
  OHigh <- ASBINOptionValue(B)
  Low <- Original - Change
  B$Volatility <- Low
  OLow <- ASBINOptionValue(B)
  B$Volatility <- Original
  OptionVega <- (OHigh - OLow)/(High - Low)
  return( OptionVega )
}
# Rho
ASBINOptionRho <- function(B){
  Original <- B$InterestRate
  Change <- (B$GreekIncrement/100.0)*Original
  High <- Original + Change
  B$InterestRate <- High
  OHigh <- ASBINOptionValue(B)
  Low <- Original - Change
  B$InterestRate <- Low
  OLow <- ASBINOptionValue(B)
  B$InterestRate <- Original
  OptionRho <- (OHigh - OLow)/(High - Low)
  return( OptionRho )
}
```