Module 8.5: Static Risk Measures
Geometric Brownian Motion-Based
Compound Option Valuation Models
R Commentary

See *Ch SRM Compound Options*.

R code for analytic delta
Very difficult to derive, hard to code, but runs fast.

```
# CO Delta
CODelta <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B12q <- exp(-q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    B12d <- exp(-d*(TU - TC))
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    mean1 <- rep(0,2)
    lower1 <- rep(-Inf,2)
    corr1 <- diag(2)
    corr1[lower.tri(corr1)] <- iC*sqrt(TC/TU)
    corr1[upper.tri(corr1)] <- iC*sqrt(TC/TU)
    upper1 <- c(iC*iU*d11,iU*d12)
    N2d11d12 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    Delta <- iC*iU*B12Nq*B2d*N2d11d12
    # Delta <- iC*iU*B1q*B12d*N2d11d12
    return(Delta)
  })
}
```

R code for numerical delta
Much easier to code compared to analytic delta, it will take longer to run and may be unstable.

```
#
# Numeric Greeks
#
# Compound option delta
CONGDelta <- function(C, L, U){
  with(C, {
    # Increment <- 0.01
    Original <- S
    Change <- Increment*Original
    High <- Original + Change
    C$S <- High
    OHigh <- COValue(C, L, U)
    Low <- Original - Change
    C$S <- Low
    OLow <- COValue(C, L, U)
    C$S <- Original
    Answer <- (OHigh - OLow)/(High - Low)
```

```
    return(Answer)
  })
}
```

GBM CO Greeks.R

```
# GBM CO Greeks.R
# rmarkdown::render("GBM CO 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
# Libraries: pracma: integral2
Packages <- c("pracma", "mvtnorm", "pbivnorm")
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)
par(family = 'Times New Roman') # Globally set fonts for graphs
#
# List expected as input in SpreadOptionValue
#
inputiC <- 1                              # 1-CO call, -1-CO put
inputiU <- 1                              # 1-UO call, -1-UO put
inputUnderlying <- 100.0                  # Must be positive
inputUnderlyingStrikePrice <- 50.0      # Must be positive
inputCompoundStrikePrice <- 50.0          # Must be positive
inputInterestRate <- 5.0 #log(1.1)*100                 # Must be positive
inputUnderlyingYield <- 0.0             # Payout of underlying instrument
inputOptionYield <- 0.0 #log(1.05)*100                    # Payout of option
inputVolatility <- 30.0                   # Must be positive
inputUnderlyingTimeToMaturity <- 1.00001     # Must be positive
inputCompoundTimeToMaturity <- 1.0        # Must be positive
inputGreekIncrement = 0.001   # In percent of greek underlying variable

COInputData <- list(inputiC, inputiU, inputUnderlying,
  inputUnderlyingStrikePrice, inputCompoundStrikePrice, inputInterestRate,
  inputUnderlyingYield, inputOptionYield, inputVolatility,
  inputUnderlyingTimeToMaturity, inputCompoundTimeToMaturity,
  inputGreekIncrement)
names(COInputData) <- c("iC", "iU", "S", "XU", "XC", "r", "d", "q", "v",
  "TU", "TC", "Increment")
# source("BSMOVM and Extended Functions.R")
source("GBM COVM w Greeks Functions.R")
# Boundaries for critical stock price iterative search
LowerBound <- 0
UpperBound <- 5.0*COInputData$S #Critical stock price for put has 2 solutions
# Tests
inputUnderlying <- COInputData$S
inputUnderlyingStrikePrice <- COInputData$XU
inputInterestRate <- COInputData$r
inputUnderlyingYield <- COInputData$d
inputOptionYield <- COInputData$q
inputVolatility <- COInputData$v
inputTimeToMaturity <- COInputData$TU
inputType <- COInputData$iU
BSMInputData <- list(inputUnderlying, inputUnderlyingStrikePrice,
  inputInterestRate, inputUnderlyingYield,  inputOptionYield,
  inputVolatility, inputTimeToMaturity, inputType)
names(BSMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
  "DividendYield", "OptionYield", "Volatility", "TimeToMaturity", "Type")
UCallValue <- BSMOptionValue(BSMInputData)
BSMInputData$Type <- -1
UPutValue <- BSMOptionValue(BSMInputData)
```

```
UCallValue; UPutValue
# Base case
COInputData$iC <- 1
COInputData$iU <- 1
CoCCSP <- COCriticalStockPrice(COInputData, UCallValue, LowerBound, UpperBound)
CoC <- COValue(COInputData, LowerBound, UpperBound)
d11 <- COd11(COInputData, LowerBound, UpperBound)
d12 <- COd12(COInputData)
d21 <- COd21(COInputData, LowerBound, UpperBound)
d22 <- COd22(COInputData)
mean1 <- rep(0,2)
lower1 <- rep(-Inf,2)
corr1 <- diag(2)
corr1[lower.tri(corr1)] <- COInputData$iC*sqrt(COInputData$TC/COInputData$TU)
corr1[upper.tri(corr1)] <- COInputData$iC*sqrt(COInputData$TC/COInputData$TU)
upper1 <- c(COInputData$iC*COInputData$iU*d11, COInputData$iU*d12)
N2d11d12 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
upper1 <- c(COInputData$iC*COInputData$iU*d21, COInputData$iU*d22)
N2d21d22 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
N1d21 <- NAp(d21)
BSMInputData$Type <- 1
BSMInputData$StrikePrice <- COInputData$XU + COInputData$XC
C <- BSMOptionValue(BSMInputData)
d1a <- d1(BSMInputData)
d2a <- d2(BSMInputData)
Nd1 <- NAp(d1a)
Nd2 <- NAp(d2a)
d11; d12; d21; d22
d1a; d2a
N2d11d12; N2d21d22; N1d21
Nd1; Nd2
CoC; C
Part1 <- COInputData$S*N2d11d12
Part2 <- -COInputData$XU*exp(-(COInputData$r/100.0)*COInputData$TU)*N2d21d22
Part3 <- -COInputData$XC*exp(-(COInputData$r/100.0)*COInputData$TC)*N1d21
SumCoC <- Part1 + Part2 + Part3
Part1; Part2; Part3; SumCoC


CoCNGDelta <- CONGDelta(COInputData, LowerBound, UpperBound)
CoCNGGamma <- CONGGamma(COInputData, LowerBound, UpperBound)
CoCNGTheta <- CONGTheta(COInputData, LowerBound, UpperBound)
COInputData$iU <- -1
CoPCSP <- COCriticalStockPrice(COInputData, UPutValue, LowerBound, UpperBound)
CoP <- COValue(COInputData, LowerBound, UpperBound)
CoPNGDelta <- CONGDelta(COInputData, LowerBound, UpperBound)
CoPNGGamma <- CONGGamma(COInputData, LowerBound, UpperBound)
CoPNGTheta <- CONGTheta(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- 1
PoCCSP <- COCriticalStockPrice(COInputData, UCallValue, LowerBound, UpperBound)
PoC <- COValue(COInputData, LowerBound, UpperBound)
PoCNGDelta <- CONGDelta(COInputData, LowerBound, UpperBound)
PoCNGGamma <- CONGGamma(COInputData, LowerBound, UpperBound)
PoCNGTheta <- CONGTheta(COInputData, LowerBound, UpperBound)
COInputData$iU <- -1
PoPCSP <- COCriticalStockPrice(COInputData, UPutValue, LowerBound, UpperBound)
PoP <- COValue(COInputData, LowerBound, UpperBound)
PoPNGDelta <- CONGDelta(COInputData, LowerBound, UpperBound)
PoPNGGamma <- CONGGamma(COInputData, LowerBound, UpperBound)
PoPNGTheta <- CONGTheta(COInputData, LowerBound, UpperBound)
CoCCSP; CoPCSP; PoCCSP; PoPCSP
CoC; CoP; PoC; PoP
CoCNGDelta; CoPNGDelta; PoCNGDelta; PoPNGDelta
```

```
CoCNGGamma; CoPNGGamma; PoCNGGamma; PoPNGGamma


# Lower Bounds
COInputData$iC <- 1
COInputData$iU <- 1
CoCLB <- COLowerBound(COInputData)
COInputData$iC <- 1
COInputData$iU <- -1
CoPLB <- COLowerBound(COInputData)
COInputData$iC <- -1
COInputData$iU <- 1
PoCLB <- COLowerBound(COInputData)
COInputData$iC <- -1
COInputData$iU <- -1
PoPLB <- COLowerBound(COInputData)
CoCLB; CoPLB; PoCLB; PoPLB
# Upper Bounds
COInputData$iC <- 1
COInputData$iU <- 1
CoCUB <- COUpperBound(COInputData)
COInputData$iC <- 1
COInputData$iU <- -1
CoPUB <- COUpperBound(COInputData)
COInputData$iC <- -1
COInputData$iU <- 1
PoCUB <- COUpperBound(COInputData)
COInputData$iC <- -1
COInputData$iU <- -1
PoPUB <- COUpperBound(COInputData)
CoCUB; CoPUB; PoCUB; PoPUB
# Delta
COInputData$iC <- 1
COInputData$iU <- 1
CoCDelta <- CODelta(COInputData, LowerBound, UpperBound)
COInputData$iC <- 1
COInputData$iU <- -1
CoPDelta <- CODelta(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- 1
PoCDelta <- CODelta(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- -1
PoPDelta <- CODelta(COInputData, LowerBound, UpperBound)
# Gamma
COInputData$iC <- 1
COInputData$iU <- 1
CoCGamma <- COGamma(COInputData, LowerBound, UpperBound)
COInputData$iC <- 1
COInputData$iU <- -1
CoPGamma <- COGamma(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- 1
PoCGamma <- COGamma(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- -1
PoPGamma <- COGamma(COInputData, LowerBound, UpperBound)
# Theta
COInputData$iC <- 1
COInputData$iU <- 1
CoCTheta <- COTheta(COInputData, LowerBound, UpperBound)
COInputData$iC <- 1
COInputData$iU <- -1
CoPTheta <- COTheta(COInputData, LowerBound, UpperBound)
```

```
COInputData$iC <- -1
COInputData$iU <- 1
PoCTheta <- COTheta(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- -1
PoPTheta <- COTheta(COInputData, LowerBound, UpperBound)

CoC; CoP; PoC; PoP
CoCLB; CoPLB; PoCLB; PoPLB
CoCUB; CoPUB; PoCUB; PoPUB
CoCCSP; CoPCSP; PoCCSP; PoPCSP
CoCDelta; CoPDelta; PoCDelta; PoPDelta
CoCNGDelta; CoPNGDelta; PoCNGDelta; PoPNGDelta
CoCGamma; CoPGamma; PoCGamma; PoPGamma
CoCNGGamma; CoPNGGamma; PoCNGGamma; PoPNGGamma

xCoC <- (COInputData$r/100)*CoC -
  (COInputData$r/100 - COInputData$q/100)*CoCDelta*COInputData$S -
  0.5*(COInputData$v/100)^2*COInputData$S*CoCGamma
xCoP <- (COInputData$r/100)*CoP -
  (COInputData$r/100 - COInputData$q/100)*CoPDelta*COInputData$S -
  0.5*(COInputData$v/100)^2*COInputData$S*CoPGamma
xPoC <- (COInputData$r/100)*PoC -
  (COInputData$r/100 - COInputData$q/100)*PoCDelta*COInputData$S -
  0.5*(COInputData$v/100)^2*COInputData$S*PoCGamma
xPoP <- (COInputData$r/100)*PoP -
  (COInputData$r/100 - COInputData$q/100)*PoPDelta*COInputData$S -
  0.5*(COInputData$v/100)^2*COInputData$S*PoPGamma

CoCTheta; CoPTheta; PoCTheta; PoPTheta
CoCNGTheta; CoPNGTheta; PoCNGTheta; PoPNGTheta
xCoC; xCoP; xPoC; xPoP
CoC; CoP; PoC; PoP

# Vega
COInputData$iC <- 1
COInputData$iU <- 1
CoCVega <- COVega(COInputData, LowerBound, UpperBound)
CoCNGVega <- CONGVega(COInputData, LowerBound, UpperBound)
COInputData$iC <- 1
COInputData$iU <- -1
CoPVega <- COVega(COInputData, LowerBound, UpperBound)
CoPNGVega <- CONGVega(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- 1
PoCVega <- COVega(COInputData, LowerBound, UpperBound)
PoCNGVega <- CONGVega(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- -1
PoPVega <- COVega(COInputData, LowerBound, UpperBound)
PoPNGVega <- CONGVega(COInputData, LowerBound, UpperBound)

CoCVega; CoPVega; PoCVega; PoPVega
CoCNGVega; CoPNGVega; PoCNGVega; PoPNGVega

# Rho
COInputData$iC <- 1
COInputData$iU <- 1
CoCRho <- CORho(COInputData, LowerBound, UpperBound)
CoCNGRho <- CONGRho(COInputData, LowerBound, UpperBound)
COInputData$iC <- 1
COInputData$iU <- -1
CoPRho <- CORho(COInputData, LowerBound, UpperBound)
CoPNGRho <- CONGRho(COInputData, LowerBound, UpperBound)
```

```
COInputData$iC <- -1
COInputData$iU <- 1
PoCRho <- CORho(COInputData, LowerBound, UpperBound)
PoCNGRho <- CONGRho(COInputData, LowerBound, UpperBound)
COInputData$iC <- -1
COInputData$iU <- -1
PoPRho <- CORho(COInputData, LowerBound, UpperBound)
PoPNGRho <- CONGRho(COInputData, LowerBound, UpperBound)

CoCRho; CoPRho; PoCRho; PoPRho
CoCNGRho; CoPNGRho; PoCNGRho; PoPNGRho
```

GBM COVM w Greeks Functions.R

```
# GBM COVM w Greeks Functions.R
#
# Present value function for BSMOVM-related calculations
PV1 = function(Maturity, Rate){
  return(exp( -(Rate/100.0) * Maturity) )
}
# n - probability density funcetion 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]) )
}
# NAp - approximation of N
NAp = function(D){
  LengthD = length(D)
  CN = numeric(LengthD)
  for(i in 1:LengthD){
    if(is.na(D[i]))D[i] = -10
    CN[i] = -99
    if(D[i] > 7) CN[i] = 1.0
    if(D[i] < -7) CN[i] = 0.0
  }
  for(j in 1:LengthD){
    if(CN[j] < 0){
      CN[j] = 0.0
      for(i in 0:12){
        CN[j]=CN[j] + exp(-( (i+0.5)^2 )/9) * sin(abs(D[j])*(i+0.5) *
          sqrt(2)/3)*(i+0.5)^(-1)
      }
      CN[j] = 0.5 + (1/pi)*CN[j]
      if(D[j] < 0) CN[j] = 1.0 - CN[j]
    }
  }
  return(CN)
}
# d - functions used in BBMOVM
d1 = function(B){
  with(B, {
    Num = ( ((InterestRate - DividendYield) / 100) + ((Volatility/100)^2)/2)*
      TimeToMaturity
    Num = log(StockPrice/StrikePrice) + Num
    Den = (Volatility/100)*sqrt(TimeToMaturity)
    return( Num/Den )
  })
}

d2 = function(B){
```

```
  with(B, {
    return( d1(B) - (Volatility/100)*sqrt(TimeToMaturity) )
  })
}
#
# BSMOVM
#
BSMOptionValue = function(B){
  with(B, {
    OptionValue = Type * StockPrice * PV1(TimeToMaturity, DividendYield - OptionYield) *
      NAp(Type * d1(B)) -
      Type * StrikePrice * PV1(TimeToMaturity, InterestRate - OptionYield) *
      NAp(Type * d2(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 )
  })
}
# Implied stock price function: May need to pass upper bound as input
BSMDYOptionImpliedStockPrice <- function(B, inputOptionValue,
  LowerBound, UpperBound){
  TestFunctionBSMDYOptionStockPrice <- function(testStockPrice, B,
    inputOptionValue){
    B$StockPrice = testStockPrice
    return( abs(inputOptionValue - BSMOptionValue(B))^2 )
  }
  solution = optimize(TestFunctionBSMDYOptionStockPrice, B, inputOptionValue,
    interval = c(LowerBound, UpperBound), tol = .Machine$double.eps^0.25)
  ImpliedStockPrice = solution$minimum
  B$StockPrice = ImpliedStockPrice
  Difference = inputOptionValue - BSMOptionValue(B)
  if (abs(Difference) < 0.01 )return(ImpliedStockPrice)
  else return(NA)
}
# Critical stock price for compound option
COCriticalStockPrice <- function(C, UOV, L, U){
  with(C, {
    inputUnderlying <- S
    inputUnderlyingStrikePrice <- XU
    inputInterestRate <- r
    inputUnderlyingYield <- d
    inputOptionYield <- q
    inputVolatility <- v
    inputTimeToMaturity <- TU - TC
    inputType <- iU
    BSMInputData <- list(inputUnderlying, inputUnderlyingStrikePrice,
      inputInterestRate, inputUnderlyingYield,  inputOptionYield,
      inputVolatility, inputTimeToMaturity, inputType)
    names(BSMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
      "DividendYield", "OptionYield", "Volatility", "TimeToMaturity", "Type")
    ISPTest <- NA # Not sure why ISP is NA for high UpperBound
    Counter <- 0
    while(is.na(ISPTest)){
```

```
      Counter <- Counter + 1.0
      ISPTest <- BSMDYOptionImpliedStockPrice(BSMInputData, UOV, L, U)
      U <- U/1.25
      if(Counter>10)break
    }
    return(ISPTest)
  })
}
# d21
COd21 <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    v <- v/100
    CSP <- COCriticalStockPrice(C, XC, L, U)
    B1Nrd <- exp((r - d)*TC)
    v1 <- v*sqrt(TC)
    d21 <- (log( (S*B1Nrd) /CSP) - ( (v1^2)/2 ))/v1
    return(d21)
  })
}
# d11
COd11 <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    v <- v/100
    CSP <- COCriticalStockPrice(C, XC, L, U)
    B1Nrd <- exp((r - d)*TC)
    v1 <- v*sqrt(TC)
    d11 <- (log( (S*B1Nrd) /CSP) + ((v1^2)/2) )/v1
    return(d11)
  })
}
# d22
COd22 <- function(C){
  with(C, {
    r <- r/100
    d <- d/100
    v <- v/100
    B2Nrd <- exp((r - d)*TU)
    v2 <- v*sqrt(TU)
    d22 <- (log( (S*B2Nrd) /XU) - ((v2^2)/2) )/v2
    return(d22)
  })
}
# d12
COd12 <- function(C){
  with(C, {
    r <- r/100
    d <- d/100
    v <- v/100
    B2Nrd <- exp((r - d)*TU)
    v2 <- v*sqrt(TU)
    d12 <- (log( (S*B2Nrd) /XU) + ((v2^2)/2) )/v2
    return(d12)
  })
}
# Compound option value
COValue <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
```

```
    v <- v/100
    B2d <- exp(-d*TU)
    B12q <- exp(-q*(TU - TC))
    B12Nq <- exp(q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B2q <- exp(-q*TU)
    B12d <- exp(-d*(TU - TC))
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    mean1 <- rep(0,2)
    lower1 <- rep(-Inf,2)
    corr1 <- diag(2)
    corr1[lower.tri(corr1)] <- iC*sqrt(TC/TU)
    corr1[upper.tri(corr1)] <- iC*sqrt(TC/TU)
    upper1 <- c(iC*iU*d11,iU*d12)
    N2d11d12 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    upper1 <- c(iC*iU*d21, iU*d22)
    N2d21d22 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    CO <- iC*iU*S*B12Nq*B2d*N2d11d12
    CO <- CO - iC*iU*XU*B12Nq*B2r*N2d21d22 - iC*XC*B1r*NAp(iC*iU*d21)
    return(CO)
  })
}
# Compound option lower bound
COLowerBound <- function(C){
  with(C, {
    inputUnderlying <- S
    inputUnderlyingStrikePrice <- XU
    inputInterestRate <- r
    inputUnderlyingYield <- d
    inputOptionYield <- q
    inputVolatility <- v
    inputTimeToMaturity <- TU
    inputType <- iU
    BSMInputData <- list(inputUnderlying, inputUnderlyingStrikePrice,
      inputInterestRate, inputUnderlyingYield, inputOptionYield,
      inputVolatility, inputTimeToMaturity, inputType)
    names(BSMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
      "DividendYield", "OptionYield", "Volatility", "TimeToMaturity", "Type")
    UOption <- BSMOptionValue(BSMInputData)
    r <- r/100
    q <- q/100
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    if(iC == 1)LB <- pmax(0, B1q*UOption - XC*B1r)
    if(iC == -1)LB <- pmax(0,XC*B1r -  B1q*UOption)
    return(LB)
  })
}

# Compound option upper bound
COUpperBound <- function(C){
  with(C, {
    inputUnderlying <- S
    inputUnderlyingStrikePrice <- XU
    inputInterestRate <- r
    inputUnderlyingYield <- d
    inputOptionYield <- q
    inputVolatility <- v
    inputTimeToMaturity <- TU
    inputType <- iU
```

```
    BSMInputData <- list(inputUnderlying, inputUnderlyingStrikePrice,
      inputInterestRate, inputUnderlyingYield, inputOptionYield,
      inputVolatility, inputTimeToMaturity, inputType)
    names(BSMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
      "DividendYield", "OptionYield", "Volatility", "TimeToMaturity", "Type")
    UOption <- BSMOptionValue(BSMInputData)
    r <- r/100
    q <- q/100
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    if(iC == 1)UB <- B1q*UOption
    if(iC == -1)UB <- XC*B1r
    return(UB)
  })
}
# CO Delta
CODelta <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B12q <- exp(-q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    B12d <- exp(-d*(TU - TC))
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    mean1 <- rep(0,2)
    lower1 <- rep(-Inf,2)
    corr1 <- diag(2)
    corr1[lower.tri(corr1)] <- iC*sqrt(TC/TU)
    corr1[upper.tri(corr1)] <- iC*sqrt(TC/TU)
    upper1 <- c(iC*iU*d11,iU*d12)
    N2d11d12 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    Delta <- iC*iU*B12Nq*B2d*N2d11d12
    # Delta <- iC*iU*B1q*B12d*N2d11d12
    return(Delta)
  })
}
# CO Gamma
COGamma <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B12d <- exp(-d*(TU - TC))
    B12Nq <- exp(q*(TU - TC))
    B1q <- exp(-q*TC)
    B2d <- exp(-d*TU)
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    rho <- sqrt(TC/TU)
    Gamma <- (B12Nq*B2d/S) *
      ( ((NAp(iU*(d12 - rho*d11)/sqrt(1-rho^2))*n(d11))/(v*sqrt(TC))) +
      (iC*(NAp(iC*iU*(d11 - rho*d12)/sqrt(1-rho^2))*n(d12))/(v*sqrt(TU))) )
```

```
    return(Gamma)
  })
}
# Compound option theta
COTheta <- function(C, L, U){
  with(C, {
        # C <- COInputData
        # L <- LowerBound
        # U <- UpperBound
        # attach(C)
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B12d <- exp(-d*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    mean1 <- rep(0,2)
    lower1 <- rep(-Inf,2)
    corr1 <- diag(2)
    corr1[lower.tri(corr1)] <- iC*sqrt(TC/TU)
    corr1[upper.tri(corr1)] <- iC*sqrt(TC/TU)
    upper1 <- c(iC*iU*d11,iU*d12)
    N2d11d12 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    upper1 <- c(iC*iU*d21, iU*d22)
    N2d21d22 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    rho <- sqrt(TC/TU)
    Theta <- -(v/2)*B12Nq*B2d *
      ( ((NAp(iU*(d12 - rho*d11)/sqrt(1 - rho^2))*n(d11))/sqrt(TC)) +
      (iC*(NAp(iC*iU*(d11 - rho*d12)/sqrt(1 - rho^2))*n(d12))/sqrt(TU)) ) +
      iC*iU*q*S*B12Nq*B2d*N2d11d12 - iC*iU*r*B12Nq*B2r*XU*N2d21d22 -
      iC*r*B1r*XC*NAp(iC*iU*d21)
    return(Theta)
  })
}
# CO Vega
COVega <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B12q <- exp(-q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    B12d <- exp(-d*(TU - TC))
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    rho <- sqrt(TC/TU)
    Vega <- S * B2d * B12Nq * (sqrt(TC)*
      NAp(iU * (d12 - rho*d11) / sqrt(1 - rho^2)) * n(d11) +
      iC * sqrt(TU) * NAp(iC*iU*(d11 - rho*d12)/sqrt(1 - rho^2))*n(d12) )
```

```
    return(Vega/100)
  })
}



# CO Rho
CORho <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    v <- v/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B12q <- exp(-q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*TC)
    B1q <- exp(-q*TC)
    B12d <- exp(-d*(TU - TC))
    d11 <- COd11(C, L, U)
    d21 <- COd21(C, L, U)
    d12 <- COd12(C)
    d22 <- COd22(C)
    mean1 <- rep(0,2)
    lower1 <- rep(-Inf,2)
    corr1 <- diag(2)
    corr1[lower.tri(corr1)] <- iC*sqrt(TC/TU)
    corr1[upper.tri(corr1)] <- iC*sqrt(TC/TU)
    upper1 <- c(iC*iU*d21,iU*d22)
    N2d21d22 <- pmvnorm(lower=lower1, upper=upper1, mean=mean1, corr=corr1)[1]
    Rho <- iC*iU*B2r*B12Nq*XU*TU*N2d21d22 + iC*B1r*XC*TC*NAp(iC*iU*d21)
    return(Rho/100)
  })
}


#
# Numeric Greeks
#
# Compound option delta
CONGDelta <- function(C, L, U){
  with(C, {
    # Increment <- 0.01
    Original <- S
    Change <- Increment*Original
    High <- Original + Change
    C$S <- High
    OHigh <- COValue(C, L, U)
    Low <- Original - Change
    C$S <- Low
    OLow <- COValue(C, L, U)
    C$S <- Original
    Answer <- (OHigh - OLow)/(High - Low)
    return(Answer)
  })
}
# Compound option gamma
CONGGamma <- function(C, L, U){
  with(C, {
    # Increment <- 0.01
    Original <- S
    Change <- Increment*Original
    High <- Original + Change
```

```
    C$S <- High
    OHigh <- COValue(C, L, U)
    Low <- Original - Change
    C$S <- Low
    OLow <- COValue(C, L, U)
    C$S <- Original
    OMid <- COValue(C, L, U)
    Answer <- ((OHigh - OMid) - (OMid - OLow))/(Change^2.0)
    return(Answer)
  })
}
# Compound option theta
CONGTheta = function(C, L, U){
  with(C, {
    OriginalTU = TU
    OriginalTC = TC
    Change <- Increment*OriginalTU
    HighTU = OriginalTU + Change
    HighTC = OriginalTC + Change
    C$TU = HighTU
    C$TC = HighTC
    OHigh = COValue(C, L, U)
    LowTU = OriginalTU - Change
    LowTC = OriginalTC - Change
    C$TU = LowTU
    C$TC = LowTC
    OLow = COValue(C, L, U)
    C$TU = OriginalTU
    C$TC = OriginalTC
    OptionTheta = (OHigh - OLow)/(2*Change)
    return( -OptionTheta ) # Moving time to maturity up is maturity time down (-)
  })
}
# Compound option vega
CONGVega <- function(C, L, U){
  with(C, {
    # Increment <- 0.01
    Original <- v # volatility, in percent (e.g., 20)
    Change <- Increment*Original
    High <- Original + Change
    C$v <- High
    OHigh <- COValue(C, L, U)
    Low <- Original - Change
    C$v <- Low
    OLow <- COValue(C, L, U)
    C$v <- Original
    Answer <- (OHigh - OLow)/(High - Low)
    return(Answer)
  })
}
# Compound option rho
CONGRho <- function(C, L, U){
  with(C, {
    # Increment <- 0.01
    Original <- r # volatility, in percent (e.g., 20)
    Change <- Increment*Original
    High <- Original + Change
    C$r <- High
    OHigh <- COValue(C, L, U)
    Low <- Original - Change
    C$r <- Low
    OLow <- COValue(C, L, U)
    C$r <- Original
    Answer <- (OHigh - OLow)/(High - Low)
```

```
    return(Answer)
  })
}
```