Module 5.7: Geometric Brownian Motion-Based Compound Option Valuation Models

See *Ch 5.7 Valuation Compound Option*. The test program is in *GBM CO Test.R*. The functions are contained in GBM COVM Functions.R.

GBM CO *Test.R (Selected Excerpts and Output)*

The following two functions facilitate the estimation of the critical stock price used in compound option valuation. The first function is simply generic to the GBMOVM. The second function modifies the compound option inputs to comply with the GBMOVM implied stock price function.

```
# Implied stock price function: May need to pass upper bound as input
GBMDYOptionImpliedStockPrice <- function(B, inputOptionValue,
  LowerBound, UpperBound){
  TestFunctionGBMDYOptionStockPrice <- function(testStockPrice, B,
    inputOptionValue){
    B$StockPrice = testStockPrice
    return( abs(inputOptionValue - GBMOptionValue(B))^2 )
  }
  solution = optimize(TestFunctionGBMDYOptionStockPrice, B, inputOptionValue,
    interval = c(LowerBound, UpperBound), tol = .Machine$double.eps^0.25)
  ImpliedStockPrice = solution$minimum
  B$StockPrice = ImpliedStockPrice
  Difference = inputOptionValue - GBMOptionValue(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
    GBMInputData <- list(inputUnderlying, inputUnderlyingStrikePrice,
      inputInterestRate, inputUnderlyingYield,  inputOptionYield,
      inputVolatility, inputTimeToMaturity, inputType)
    names(GBMInputData) <- c("StockPrice", "StrikePrice", "InterestRate",
      "DividendYield", "OptionYield", "Volatility", "TimeToMaturity", "Type")
    return(GBMDYOptionImpliedStockPrice(GBMInputData, UOV, L, U))
  })
}
```

The key function in GBM COVM Functions.R is the valuation function, COValue.

```
COValue <- function(C, L, U){
  with(C, {
    r <- r/100
    d <- d/100
    q <- q/100
    B2d <- exp(-d*TU)
    B12Nq <- exp(q*(TU - TC))
    B2r <- exp(-r*TU)
    B1r <- exp(-r*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)
  })
}
```