Package ‘RHestonSLV’

May 16, 2016

Type Package

Title R Implementation of the Heston Stochastic Local Volatility Model

Version 0.1.0

Author Klaus Spanderen

Maintainer <klaus@spanderen.de>

Description The RHestonSLV package makes QuantLib's implementation of the Heston Stochastic Local Volatility Model accessible from R. Local Stochastic Volatility (LSV) models have become the industry standard for FX and equity markets. The local volatility extension of the popular Heston stochastic volatility model is a promising candidate within the zoo of LSV models. But the calibration of this model is not only computational demanding but also tricky from an algorithmic point of view, especially if the Feller constraint is violated The two main solutions to tackle the calibration problem are either solving the Fokker-Planck forward equation via finite difference methods or based on efficient Monte-Carlo simulations. Pricing and greek calculations for vanilla and more exotic options are also supported.

License GPL 2.0

LazyData TRUE

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp

SystemRequirements QuantLib library (>= 1.8.0) from http://quantlib.org, Boost library from http://www.boost.org

NeedsCompilation yes

R topics documented:

`HestonLocalVolSurface` *Class* `"HestonLocalVolSurface"`

Description

Translates a Heston model into the corresponding local volatility surface with the same option prices.

Arguments

`referenceDate`	a date setting the reference date for the calculation
`maxDate`	a date setting the end date of the calculation
`hestonProcess`	an object of the class HestonProcess defining the Heston model

Objects from the Class

An instance of the class calculates the corresponding local volatility surface for a given the Heston model, which results in the same option prices. Objects can be created by calls of the form `new("HestonLocalVolSurface", referenceDate, maxDate, hestonProcess)`

Examples

```
#Heston process with r=0.05, c=0.02, spot=100, v0=0.09, kappa=1, theta=0.06, sigma=0.4 and rho=-0.75
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

hestonLocalVol <- new(HestonLocalVolSurface, Sys.Date(),Sys.Date()+365, process)

s <- seq(50, 200, 1)
plot(s, sapply(s, function(spot) { hestonLocalVol$localVol(1.0, spot) }), type='l',
     ylab="Local Volatility", xlab="Spot", main="Local Volatility in One Year")
```

HestonProcess *Class* "HestonProcess"

Description

The Heston process is give by

$$dS_t = (r_t - q_t)S_t dt + \sqrt{\nu_t} S_t dW_t^S$$
$$d\nu_t = \kappa(\theta - \nu_t)dt + \sigma\sqrt{\nu_t} dW_t^\nu$$
$$\rho dt = dW_t^S dW_t^\nu$$

Objects from the Class

An object represents the paramter of a Heston process. Objects can be created by calls of the form `new("HestonProcess", ...)` or `HestonProcess(r, q, spot, v0, kappa, theta, sigma, rho)`.

Slots

`r`: domestic interest rate as a function of time (continuous compounding, Actual365Fixed)

`q`: divident rate or foreign interest rate as a function of time (continuous compounding, Actual365Fixed)

`spot`: current price of the underlying stock or fx rate

`v0`: spot variance $\nu_{t=0}$

`kappa`: rate at which ν_t reverts to θ

`theta`: mean reversion level of the variance ν_t

`sigma`: volatility of volatility

`rho`: correlation between spot and variance increments

Examples

```
> process <- HestonProcess(function(t) { 0.02 },
                           function(t) { 0.01 },
                           100, 0.09, 2.0, 0.06, 0.4, -0.75)

> process
HestonProcess
  r(t=0):  0.02
  q(t=0):  0.01
  spot  :  100
  v0    :  0.09
  kappa :  2
  theta :  0.06
  sigma :  0.4
  rho   :  -0.75

> process["rho"]
[1] -0.75
> process["sigma"] <- 0.2
```

```
> process["sigma"]
[1] 0.2
> process["sigma"] <- -0.2
 Error in validObject(x) :
   invalid class "HestonProcess" object: negative sigma was given.
> process["rho"] <- 2
 Error in validObject(x) :
   invalid class "HestonProcess" object: correlation rho must stay between [-1,1].
```

---

hestonSLVBarrierPricer

Barrier Option Pricer for the Heston SLV Model

Description

The hestonSLVBarrierPricer function evaluates a barrier option with European exercise under the Heston Stochastic Volatility model using Finite Difference methods. The option value and the common first derivatives ("Greeks") are returned.

Usage

```
\code{hestonSLVBarrierPricer}(referenceDate, barrier, rebate, barrierType, strike, optionType,
                     maturityDate, hestonProcess, leverageFunction, tGrid=51, xGrid=401,
                      vGrid=51, dampingSteps=0, fdmScheme = "ModifiedCraigSneyd")
```

Arguments

referenceDate	a date setting the reference date for the calculation
barrier	the barrier level
rebate	rebate if barrier is knocked out
barrierType	a string with one of the values “downin”, “downout”, “upin” or “upout”
strike	the strike price of the option
optionType	a string with one of the values “call” or “put”
maturityDate	the maturity date of the barrier option
hestonProcess	the Heston model part of the HestonSLV specification
leverageFunction	the leverage function of the HestonSLV model
tGrid	number of time steps for the Finite Difference scheme
xGrid	number of grid points in spot direction
vGrid	number of grid points in variance direction
dampingSteps	number of damping steps to avoid spurious oscillations
fdmScheme	the Finite Difference scheme, a string with one of the values “Hundsdorfer”, “ModifiedHundsdorfer”, “Douglas”, “CraigSneyd”, “ModifiedCraigSneyd”, “ImplicitEuler” or “ExplicitEuler”

Value

The `hestonSLVBarrierPricer` function returns a list with the following components:

`value`	npv of option
`delta`	change in option value for a change in the underlying
`gamma`	change in option delta for a change in the underlying
`theta`	change in option value for a change in t
`impliedVol`	implied Black-Scholes-Merton volatility of the option

defined

Examples

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

leverageFct <- function(t, s) { exp(-t)*(s+70)/100.0 }

b <- seq(50, 100, 5)
plot(b, sapply(b, function(barrier) {
  hestonSLVBarrierPricer(Sys.Date(), barrier, 0.0, "downout", 100, "put",
                        Sys.Date()+365, process, leverageFct)$gamma
  }), type="b",lty=2, ylab="NPV",xlab="Strike"
)
```

`hestonSLVDoubleNoTouchBarrierPricer`
Double No Touch Barrier Option Pricer for the Heston SLV Model

---

Description

The `hestonSLVDoubleNoTouchBarrierPricer` function evaluates a double-no-touch barrier option with European exercise under the Heston Stochastic Volatility model using Finite Difference methods. The option value and the common first derivatives ("Greeks") are returned.

Usage

```
\code{hestonSLVDoubleNoTouchBarrierPricer}(
    referenceDate, barrier_lo, barrier_hi, rebate, barrierType, strike, optionType,
    payoffType, maturityDate, hestonProcess, leverageFunction, tGrid=51, xGrid=401,
      vGrid=51, dampingSteps=0, fdmScheme = "ModifiedCraigSneyd")
```

Arguments

referenceDate	a date setting the reference date for the calculation
barrier_lo	the lower barrier
barrier_lo	the upper barrier
rebate	rebate if barrier is knocked out
barrierType	a string with one of the values “KnockIn”, “KnockOut”, “KIKO” or “KOKI”
strike	the strike price of the option
optionType	a string with one of the values “call” or “put”
payofftype	a string with one of the values “PlainVanilla”, “CashOrNothing” or “AssetOrNothing”
maturityDate	the maturity date of the barrier option
hestonProcess	the Heston model part of the HestonSLV specification
leverageFunction	the leverage function of the HestonSLV model
tGrid	number of time steps for the Finite Difference scheme
xGrid	number of grid points in spot direction
vGrid	number of grid points in variance direction
dampingSteps	number of damping steps to avoid spurious oscillations
fdmScheme	the Finite Difference scheme, a string with one of the values “Hundsdorfer”, “ModifiedHundsdorfer”, “Douglas”, “CraigSneyd”, “ModifiedCraigSneyd”, “ImplicitEuler” or “ExplicitEuler”

Value

The hestonSLVDoubleNoTouchBarrierPricer function returns a list with the following components:

value	value of option

Examples

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

leverageFct <- function(t, s) { exp(-t)*(s+70)/100.0 }

s <- seq(50, 150, 5)
plot(s, sapply(s, function(spot) {
  process["spot"] <- spot
  hestonSLVDoubleNoTouchBarrierPricer(Sys.Date(), 50, 150, 0.0, "KnockOut", 0, "call",
                                      "CashOrNothing", Sys.Date()+365, process,
                                      leverageFct)$value
  }), type="b",lty=2, ylab="NPV",xlab="Strike"
)
```

HestonSLVFDMModel *Class* "HestonSLVFDMModel"

Description

Calibration of the Heston Stochastic Local Volatility model

$$\begin{aligned} dx_t &= \left(r_t - q_t - \frac{L^2(t,x_t)}{2}\nu_t\right)dt + L(t,x_t)\sqrt{\nu_t}dW_t^x \\ d\nu_t &= \kappa\left(\theta - \nu_t\right)dt + \eta\sigma\sqrt{\nu_t}dW_t^\nu \\ \rho dt &= dW_t^\nu dW_t^x \end{aligned}$$

via Finite Difference methods solving the Fokker-Planck forward equation.

Arguments

`referenceDate` a date setting the reference date for the calibration

`maxCalibrationDate` a date setting the end date of the calibration

`localVolFunction` a function in (time, underlying) defining the local volatility function. The HestonSLVMCModel calculates the leverage function such that the Heston SLV model defined by the Heston process and the leverage function gives the same prices as the local volatility model.

`hestonProcess` an object of the class HestonProcess

`hestonSLVFDMParams` an object of the class HestonSLVFDMParams

Objects from the Class

An object of this class calibrates a Heston Stochastic Local Volatility model to a given local volatility model w.r.t. a Heston process. Objects can be created by calls of the form
`new("HestonSLVFDMModel", referenceDate, maxCalibrationDate,`
`localVolFunction, hestonProcess, hestonSLVFMParams)`

References

A.Stoep, L. Grzelak, C. Oosterlee, The Heston Stochastic-Local Volatility Model: Efficient Monte Carlo Simulation, http://ta.twi.tudelft.nl/mf/users/oosterle/oosterlee/anton1.pdf
Johannes Goettker-Schnetmann, Klaus Spanderen, Calibrating the Heston Stochastic Local Volatility Model using the Fokker-Planck Equation, http://hpc-quantlib.de/src/slv.pdf

Examples

```
#flat local volatility surface
localVol <- function(t, s) { 0.3 }

#Heston process with r=0.05, c=0.02, spot=100, v0=0.09, kappa=1, theta=0.06, sigma=0.4 and rho=-0.75
```

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

params <- new ("HestonSLVFDMParams")

# calibrate HestonSLV model for the next year
fdmModel <- new (HestonSLVFDMModel,
                Sys.Date(), Sys.Date()+365,
                localVol, process, params)

s <- seq(50, 200, 1)
plot(s, sapply(s, function(spot) { fdmModel$leverageFunction(1.0, spot) }), type='l',
     ylab="Leverage Function", xlab="Spot", main="Leverage Function in One Year")
```

HestonSLVFDMParams   *Class* "HestonSLVFDMParams"

Description

Defines the parameter for a calibration of the Heston Stochastic Local Volatility model via the Fokker-Planck forward equation.

Objects from the Class

An object of this class defines the parameter for a calibration based on the Fokker-Planck equation via Finite Difference methods. Objects can be created by calls of the form `new("HestonSLVFDMParams", ...)` or `HestonSLVFDMParams(...)`.

Slots

`xGrid`: number of the grid points in spot direction

`vGrid`: number of the grid points in variance direction

`tMaxStepsPerYear`: maximum number of time steps per year

`tMinStepsPerYear`: minimum number of time steps per year

`tStepNumberDecay`: rate of decay for time steps per year

`predictionCorrectionSteps`: number of prediction/correction steps

`x0Density`: density factor at origin for mesher in spot direction

`localVolEpsProb`: mesher stopping condition in spot direction

`maxIntegrationIterations`: maximum number of integration steps

`vLowerEps`: mesher stopping condition in variance direction for the lower bound

`vUpperEps`: mesher stopping condition in variance direction for the upper bound

`vMin`: lower bound for the mesher in variance direction

`v0Density`: density factor for mesher in variance direction at the origin

`vLowerBoundDensity`: density factor for mesher in variance direction at the lower bound `vUpperBoundDensity`: density factor for mesher in variance direction at the upper bound

`leverageFctPropEps`: extrapolate leverage function if probability is below this value

`greensAlgorithm`: algorithm for the approximation of the Greens function at t=0. ("ZeroCorrelation", "Gaussian" or "SemiAnalytical")

`transformationType`: coordinate transformation in spot direction ("Plain", "Power" or "Log")

`fdmSchemeType`: discretization scheme in time direction ("Hundsdorfer", "ModifiedHundsdorfer", "Douglas", "CraigSneyd", "ModifiedCraigSneyd", "ImplicitEuler" or "ExplicitEuler")

References

Johannes Goettker-Schnetmann, Klaus Spanderen http://hpc-quantlib.de/src/slv.pdf Calibrating the Heston Stochastic Local Volatility Model using the Fokker-Planck Equation

Examples

```
> params <- new ("HestonSLVFDMParams")
> params
HestonSLVFDMParams
  xGrid                     :  301
  vGrid                     :  601
  tMaxStepsPerYear          :  2000
  tMinStepsPerYear          :  30
  tStepNumberDecay          :  2
  predictionCorrectionSteps:  2
  x0Density                 :  0.1
  localVolEpsProb           :  1e-04
  maxIntegrationIterations :  10000
  vLowerEps                 :  1e-05
  vUpperEps                 :  1e-05
  vMin                      :  2.5e-06
  v0Density                 :  1
  vLowerBoundDensity        :  0.1
  vUpperBoundDensity        :  0.9
  leverageFctPropEps        :  1e-05
  greensAlgorithm           :  Gaussian
  transformationType        :  Log
  fdmSchemeType             :  ModifiedCraigSneyd

> params["fdmSchemeType"] <- "Hundsdorfer"
> params["localVolEpsProb"] <- 1e-6
> params["greensAlgorithm"] <- "ZeroCorrelation"
> params["greensAlgorithm"]
[1] "ZeroCorrelation"
```

`hestonSLVForwardOptionPricer` *Vanilla Forward Starting Option Pricer for the Heston SLV Model*

Description

The hestonSLVForwardOptionPricer function evaluates a forward starting vanilla european option under the Heston Stochastic Volatility model using Monte-Carlo simulations. The option value and the implied Black-Scholes-Merton volatility are returned.

Usage

```
\code{hestonSLVForwardOptionPricer}(referenceDate, strike, resetDate, optionType, maturityDate,
                                    nSimulations, hestonProcess, leverageFunction)
```

Arguments

referenceDate	a date setting the reference date for the calculation
strike	the relative strike of the option. The absolute strike is spot value at reset date times the relative strike
resetDate	the reset date at which the absolute strike is fixed
optionType	a string with one of the values “call” or “put”
maturityDate	the maturity date
nSimulation	number of Monte-Carlo simulations used to calculate the option value
hestonProcess	the Heston model part of the HestonSLV specification
leverageFunction	the leverage function of the HestonSLV model

Value

The hestonSLVForwardOptionPricer function returns a list with the following components:

value	Value of option
impliedVol	implied Black-Scholes-Merton volatility of the option

Examples

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

leverageFct <- function(t, s) { exp(-t)*(s+70)/100.0 }

s <- seq(0.7, 1.5, 0.1)
plot(s, sapply(s, function(strike) {
  hestonSLVForwardOptionPricer(Sys.Date(), strike, Sys.Date()+182, "call", Sys.Date()+365,
                               4000, process, leverageFct)$impliedVol
                }), type="b",lty=2, ylab="Implied Volatility",xlab="Strike"
)
```

`HestonSLVMCModel` *Class* `"HestonSLVMCModel"`

Description

Calibration of the Heston Stochastic Local Volatility model

$$\begin{aligned} dx_t &= \left(r_t - q_t - \frac{L^2(t,x_t)}{2}\nu_t\right)dt + L(t,x_t)\sqrt{\nu_t}dW_t^x \\ d\nu_t &= \kappa\left(\theta - \nu_t\right)dt + \eta\sigma\sqrt{\nu_t}dW_t^\nu \\ \rho dt &= dW_t^\nu dW_t^x \end{aligned}$$

via Monte-Carlo simulations.

Arguments

`referenceDate` a date setting the reference date for the calibration

`maxCalibrationDate` a date setting the end date of the calibration

`localVolFunction` a function in (time, underlying) defining the local volatility function. The HestonSLVMCModel calculates the leverage function such that the Heston SLV model defined by the Heston process and the leverage function gives the same prices as the local volatility model.

`hestonProcess` an object of the class HestonProcess

`hestonSLVMCParams` an object of the class HestonSLVMCParams

Objects from the Class

An object of this class calibrates a Heston Stochastic Local Volatility model to a given local volatility model w.r.t. a Heston process. Objects can be created by calls of the form
`new("HestonSLVMCModel", referenceDate, maxCalibrationDate,`
`localVolFunction, hestonProcess, hestonSLVMCParams)`

References

A.Stoep, L. Grzelak, C. Oosterlee, The Heston Stochastic-Local Volatility Model: Efficient Monte Carlo Simulation, http://ta.twi.tudelft.nl/mf/users/oosterle/oosterlee/anton1.pdf
Johannes Goettker-Schnetmann, Klaus Spanderen, Calibrating the Heston Stochastic Local Volatility Model using the Fokker-Planck Equation, http://hpc-quantlib.de/src/slv.pdf

Examples

```
#flat local volatility surface
localVol <- function(t, s) { 0.3 }

#Heston process with r=0.05, c=0.02, spot=100, v0=0.09, kappa=1, theta=0.06, sigma=0.4 and rho=-0.75
```

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

params <- HestonSLVMCParams(TRUE, 90, 50, 10000)

# calibrate HestonSLV model for the next year
mcModel <- new (HestonSLVMCModel,
                Sys.Date(), Sys.Date()+365,
                localVol, process, params)

s <- seq(50, 150, 1)
plot(s, sapply(s, function(spot) { mcModel$leverageFunction(1.0, spot) }), type='l',
     ylab="Leverage Function", xlab="Spot", main="Leverage Function in One Year")
```

HestonSLVMCParams *Class* "HestonSLVMCParams"

Description

Defines the parameter for a Monte-Carlo calibration of the Heston Stochastic Local Volatility Model.

Objects from the Class

An instance of the class defines the numerical parameter for a Monte-Carlo calibration. Objects can be created by calls of the form `new("HestonSLVFDMParams", ...)` or `HestonSLVMCParams(qmc, timeStepsPerYear, nBins, calibrationPaths)`.

Slots

`qmc`: logical, defines if Sobol Quasi Monte-Carlo numbers in conjunction with Brownian bridges should be used to draw the paths.

`timeStepsPerYear`: number of time steps per year

`nBins`: number of bins in ever time step

`calibrationPaths`: number of Monte-Carlo paths to be used for the calibration

References

A.Stoep, L. Grzelak, C. Oosterlee, The Heston Stochastic-Local Volatility Model: Efficient Monte Carlo Simulation, http://ta.twi.tudelft.nl/mf/users/oosterle/oosterlee/anton1.pdf

Examples

```
> params <- HestonSLVMCParams(TRUE, 90, 100, 32767)
> params
HestonSLVMCParams
  qmc             :  TRUE
  timeStepsPerYear:  90
```

```
  nBins          :  100
  calibrationPaths:  32767

> params["qmc"]
[1] TRUE
> params["nBins"] <- 200
> params["nBins"]
[1] 200
```

---

hestonSLVOptionPricer *Vanilla Option Pricer for the Heston SLV Model*

Description

The hestonSLVOptionPricer function evaluates a vanilla option with European or American exercise under the Heston Stochastic Volatility model using Finite Difference methods. The option value and the common first derivatives ("Greeks") are returned.

Usage

```
\code{hestonSLVOptionPricer}(referenceDate, strike, optionType, exerciseType,
                  maturityDate, hestonProcess, leverageFunction, tGrid=51, xGrid=401,
                   vGrid=51, dampingSteps=0, fdmScheme = "ModifiedCraigSneyd")
```

Arguments

referenceDate	a date setting the reference date for the calculation
strike	the strike price of the option
optionType	a string with one of the values “call” or “put”
exerciseType	a string with one of the values “european” or “american”
maturityDate	the maturity date
hestonProcess	the Heston model part of the HestonSLV specification
leverageFunction	the leverage function of the HestonSLV model
tGrid	number of time steps for the Finite Difference scheme
xGrid	number of grid points in spot direction
vGrid	number of grid points in variance direction
dampingSteps	number of damping steps to avoid spurious oscillations
fdmScheme	the Finite Difference scheme, a string with one of the values “Hundsdorfer”, “ModifiedHundsdorfer”, “Douglas”, “CraigSneyd”, “ModifiedCraigSneyd”, “ImplicitEuler” or “ExplicitEuler”

Value

The `hestonSLVOptionPricer` function returns a list with the following components:

value	npv of option
delta	change in option value for a change in the underlying
gamma	change in option delta for a change in the underlying
theta	change in option value for a change in t
impliedVol	implied Black-Scholes-Merton volatility of the option

Examples

```
process <- HestonProcess(function(t,s) {0.05}, function(t,s) {0.02},
                         100, 0.09, 1.0, 0.06, 0.4, -0.75)

leverageFct <- function(t, s) { exp(-t)*(s+70)/100.0 }

s <- seq(50, 200, 10)
plot(s, sapply(s, function(strike) {
  hestonSLVOptionPricer(Sys.Date(), strike, "call", "european",
                        Sys.Date()+182, process, leverageFct)$impliedVol
                }), type="b",lty=2, ylab="Implied Volatility",xlab="Strike"
)
```

Index