Package: multiAssetOptions (via r-universe)

September 14, 2024

Type Package

Title Finite Difference Method for Multi-Asset Option Valuation

Version 0.1-2

Date 2021-04-20

Author Michael Eichenberger and Carlo Rosa

Maintainer Michael Eichenberger <mike.eichenberger@gmail.com>

Imports Matrix, graphics

Description Efficient finite difference method for valuing European and American multi-asset options.

License GPL-2 | GPL-3

NeedsCompilation no

Date/Publication 2021-04-20 14:50:07 UTC

Repository https://mike4358.r-universe.dev

RemoteUrl https://github.com/cran/multiAssetOptions

RemoteRef HEAD

RemoteSha 4d205e9c5a1014d3e257cfefdaa04e232d712d95

Contents

`multiAssetOptions-package` *Finite Difference Method for Multi-Asset Option Valuation*

Description

This package implements an efficient finite difference method for valuing multi-asset options in the Black-Scholes world. The model assumes static volatilities and correlations. The implementation allows users to vary the option setup (number of underlying assets, call vs. put, European vs. American, etc.) as well as some of the features of the numerical method (mesh spacing, timestepping scheme, etc.).

Details

Package:	multiAssetOptions
Type:	Package
Version:	0.1-2
Date:	2021-04-20
License:	GPL-2 | GPL-3

This package contains the function `multiAssetOption`, which implements a generalized version of the finite difference method for option valuation. Several sub-functions are included in this package to improve code clarity and presentation.

Author(s)

Michael Eichenberger and Carlo Rosa

Maintainer: Michael Eichenberger <mike.eichenberger@gmail.com>

`matrixFDM` *Finite Difference Matrix Generator*

Description

Generates a modified coefficient matrix (**M**-matrix) used in the finite difference method from the option inputs. See Tavella and Randall (2000) for more on the standard formulation of the **M**-matrix.

Usage

```
matrixFDM(S, rf, q, vol, rho)
```

Arguments

S	list containing the vectors of spatial grid points associated with each underlying. Vector sizes of underlying spatial grid points need not be equal.
rf	scalar; applicable risk-free rate (domestic risk-free rate).
q	vector; holding costs of the option's underlyings (dividends, foreign risk-free rates, etc.).
vol	vector; volatilities of the option's underlyings.
rho	matrix; correlation matrix of the option's underlyings.

Details

matrixFDM first constructs the non-zero diagonals of the **M**-matrix and stores them as columns. The bandSparse function from the **Matrix** package then constructs a sparse banded matrix from the columns of the previously contructed matrix. Spatial domain boundaries are calculated first-order inwards with second difference terms dropped, maintaining block tridiagonality.

Value

matrixFDM returns a CsparseMatrix-class matrix used for timestepping in the finite difference method.

Author(s)

Michael Eichenberger and Carlo Rosa

References

Tavella, D., Randall, C., 2000. Pricing Financial Instruments: The Finite Difference Method. John Wiley & Sons, Inc., New York.

Examples

```
# finite difference matrix for uniformly-spaced two-asset option
S1 <- list(seq(0, 5, by=1), seq(0, 5, by=1))
rf <- 0.1
q <- c(0.05, 0.04)
vol <- c(0.20, 0.25)
rho <- matrix(c(1,-0.5,-0.5,1), 2, 2)
matrixFDM(S1, rf, q, vol, rho)
```

`multiAssetOption` *Finite Difference Method for Multi-Asset Option Valuation*

Description

`multiAssetOption` generalizes the standard finite difference method to handle mulitple underlying assets, non-uniform grid spacing, non-uniform timestepping, and American exercise. The implementation allows users to vary the option setup (number of underlying assets, call vs. put, European vs. American, etc.) as well as the features of the numerical method (grid spacing, timestepping scheme, etc.). Strike shifting the mesh and Rannacher smoothing are optionally included to remedy problems arising from potential spurious oscillations in the solution.

Usage

```
multiAssetOption(X)
```

Arguments

`X`	list of inputs. List items given in the **Details** section.

Details

Items of the input list `X` are as follows:

`X$opt$nAsset` integer; number of underlying assets.

`X$opt$payType` case; if 0, digital payoff, if 1, best-of payoff, if 2, worst-of payoff.

`X$opt$exerType` case; if 0, European exercise, if 1, American exercise.

`X$opt$pcFlag` case vector; if 0, call, if 1, put.

`X$opt$ttm` scalar; time to maturity.

`X$opt$strike` vector; option strikes.

`X$opt$rf` scalar; applicable risk-free rate (domestic risk-free rate).

`X$opt$q` vector; holding costs of the option's underlyings (dividends, foreign risk-free rates, etc.).

`X$opt$vol` vector; volatilities of the option's underlyings.

`X$opt$rho` matrix; correlation matrix of the option's underlyings.

`X$fd$m` vector; number of spatial steps for each underlying's domain discretization.

`X$fd$leftBound` vector; near spatial boundaries of each underlying's domain.

`X$fd$kMult` vector; right boundary strike multiples. If 0, far domain boundary calculated via formula given in Kangro and Nicolaides (2000). Otherwise, far domain boundary calculated as the strike multiplied by the strike multiple.

`X$fd$density` vector; impacts the concentration of nodes around the option strike. At 0, nodes are uniformly distributed between the near and far boundaries. Increasing the parameter increases the node concentration around the strike.

`X$fd$kShift` case vector; if 0, no mesh shifting, if 1, adjusts the node spacing such that the strike falls exactly between two nodes, if 2, adjusts the node spacing such that the strike falls exactly on a node. See Tavella and Randall (2000).

`X$fd$theta` scalar; implicitness parameter of the theta method. Chosen between 0 (fully explicit) and 1 (fully implicit).

`X$fd$maxSmooth` integer; number of Rannacher smoothing steps. See Rannacher (1984).

`X$fd$tol` scalar; error tolerance in penalty iteration for American exercise.

`X$fd$maxIter` integer; maximum number of iterations per penalty loop for American exercise.

`X$time$tsType` case; if 0, constant timestep size, if 1, adaptive timestep size. See Forsyth and Vetzal (2002).

`X$time$N` integer; number of total timesteps if not using adaptive timesteps.

`X$time$dtInit` scalar; inital timestep size for adaptive timesteps.

`X$time$dNorm` scalar; target relative change for adaptive timesteps.

`X$time$D` scalar; normalizing parameter for adaptive timesteps.

The classical order for the state vectors output from the function is illustrated by example. With two underlying assets, option values in each state vector are stored in the order: [11, 21, 31, ... , M1, 12, 22, ... , MN] with M being the total number of nodes used in the first asset spatial discretization and N being the total number of nodes in the second.

Value

`multiAssetOption` returns a list:

`value`	matrix of per-unit option values. Each column stores the state of the option value array (collection of option values for all nodes of the spatial mesh) as a vector following the classical order (see **Details** section). The columns of the matrix are indexed over time, with the first column beginning at option maturity, and subsequent columns moving backward in time.
`S`	list containing the vectors of spatial grid points associated with each underlying. Vector sizes of underlying spatial grid points need not be equal.
`dimS`	dimension of option value array. This item can be used to reshape the column vectors in `value` into an appropriately dimensioned array using `array( ... , dim=dimS)`.
`time`	vector of times associated with each column of the `value` item.

For each column (time) of the `value` item, the option value at that time can be calculated as the option's notional amount multiplied by the unit option value interpolated over the `S` item at the current underlying prices.

Author(s)

Michael Eichenberger and Carlo Rosa

References

Forsyth, P.A., Vetzal, K.R., 2002. Quadratic convergence for valuing American options using a penalty method. *SIAM Journal on Scientific Computing*, **23** (6), 2095–2122.

Kangro, R., Nicolaides, R., 2000. Far field boundary conditions for Black-Scholes equations. *SIAM Journal on Numerical Analysis*, **38** (4), 1357–1368.

Rannacher, R., 1984. Finite element solution of diffusion problems with irregular data. *Numberische Mathematik*, **43**, 309–327.

Tavella, D., Randall, C., 2000. Pricing Financial Instruments: The Finite Difference Method. John Wiley & Sons, Inc., New York.

Examples

```
# european dual-asset digital option example

# initialize inputs list
X <- list()

# option inputs
X$opt$nAsset <- 2
X$opt$payType <- 0
X$opt$exerType <- 0
X$opt$pcFlag <- c(1, 0)
X$opt$ttm <- 0.5
X$opt$strike <- c(110, 90)
X$opt$rf <- 0.10
X$opt$q <- c(0.05, 0.04)
X$opt$vol <- c(0.20, 0.25)
X$opt$rho <- matrix(c(1, -0.5, -0.5, 1), X$opt$nAsset, X$opt$nAsset)

# finite difference inputs
X$fd$m <- c(20, 20)
X$fd$leftBound <- c(0, 0)
X$fd$kMult <- c(0, 0)
X$fd$density <- c(5, 5)
X$fd$kShift <- c(1, 1)
X$fd$theta <- 0.5
X$fd$maxSmooth <- 2
X$fd$tol <- 1e-7
X$fd$maxIter <- 3

# timestep inputs
X$time$tsType <- 0
X$time$N <- min(X$fd$m) * 4
X$time$dtInit <- 0.1 / 4^log2(min(X$fd$m)/5)
X$time$dNorm <- 5 / 2^log2(min(X$fd$m)/5)
X$time$D <- 0.05

# function check
output <- multiAssetOption(X)
```

nodeSpacer *Non-Uniform Finite Difference Node Spacer*

Description

nodeSpacer implements the spatial discretization scheme from Hout and Foulon (2010) with arbitrary left and right bounds. The function additionally includes logic for mesh shifting.

Usage

```
nodeSpacer(K, leftBound, rightBound, nodes, density, kShift)
```

Arguments

K	scalar; option strike.
leftBound	scalar; near spatial boundary of the underlying domain.
rightBound	scalar; far spatial boundary of the underlying domain.
nodes	integer; number of nodes used in the spatial discretization.
density	scalar; impacts the concentration of nodes around the option strike. At 0, nodes are uniformly distributed between the leftBound and rightBound. Increasing the parameter increases the node concentration around the strike.
kShift	case; if 0, no mesh shifting, if 1, adjusts the node spacing such that the strike falls exactly between two nodes, if 2, adjusts the node spacing such that the strike falls exactly on a node. See Tavella and Randall (2000).

Details

Mesh shifting is accomplished by multiplying the vector of gridpoints by a scalar. For multi-asset options, this nodeSpacer is called iteratively to discretize each underlying's spatial domain.

Value

nodeSpacer returns a vector of gridpoints used in spatial discretization in the finite difference method. The nodes input determines the length of the output vector.

Author(s)

Michael Eichenberger and Carlo Rosa

References

Hout, K. J., Foulon, S., 2010. ADI finite difference schemes for option pricing in the Heston model with correlation. *International Journal of Numerical Analysis and Modeling*, **7** (2), 303–320. http://www.math.ualberta.ca/ijnam/Volume-7-2010/No-2-10/2010-02-06.pdf

Pooley, D. M., Vetzal, K. R., Forsyth, P. A., 2002. Convergence remedies for non-smooth payoffs in option pricing. https://cs.uwaterloo.ca/~paforsyt/report.pdf

Tavella, D., Randall, C., 2000. Pricing Financial Instruments: The Finite Difference Method. John Wiley & Sons, Inc., New York.

Examples

```
# sample mesh spacing
plot(nodeSpacer(100, 0, 500, 26, 5, 1), rep(0, times=26), main='Non-Uniform Mesh Spacing',
  xlab='Underlying Price (Strike = 100)', ylab='', yaxt='n', type='p', cex=0.5, pch=16)
```

---

payoff *Multi-Asset Option Payoff Calculator*

Description

payoff calculates the per-unit option payoff for digital, best-of, and worst-of multi-asset options.

Usage

```
payoff(payType, pcFlag, strike, S)
```

Arguments

payType	case; if 0, digital payoff, if 1, best-of payoff, if 2, worst-of payoff.
pcFlag	case vector; if 0, call, if 1, put.
strike	vector; option strikes.
S	list containing the vectors of spatial grid points associated with each underlying. Vector sizes of underlying spatial grid points need not be equal.

Value

payoff returns an array of the unit option values at each point spanned by the list of underlying vectors. Dimension of array is inhereted from S.

Author(s)

Michael Eichenberger and Carlo Rosa

Examples

```
# payoff of a dual-asset digital call with strikes at 100 and 90.
S <- list(seq(0, 500, by=1), seq(0, 500, by=1))
payoff(0, c(0, 0), c(100, 90), S)
```

plotOptionValues *Plot Option Values Over Time*

Description

plotOptionValues plots the solution of the option PDE over time.

Usage

```
plotOptionValues(Y, fps)
```

Arguments

Y	list containing the items resulting from the multiAssetOption function.
fps	integer; number of frames per second of the animation.

Details

Animation occurs in backwards time, beginning from the option's expiry date, moving toward time = 0. This function applies only to options written on one or two underlying assets. Higher dimensional options are not plotted.

Author(s)

Michael Eichenberger and Carlo Rosa

See Also

multiAssetOption

Examples

```
# plot test

# initialize inputs list
X <- list()

# option inputs
X$opt$nAsset <- 2
X$opt$payType <- 0
X$opt$exerType <- 0
X$opt$pcFlag <- c(0, 0)
X$opt$ttm <- 0.5
X$opt$strike <- c(110, 90)
X$opt$rf <- 0.10
X$opt$q <- c(0.05, 0.04)
X$opt$vol <- c(0.20, 0.25)
X$opt$rho <- matrix(c(1, -0.5, -0.5, 1), X$opt$nAsset, X$opt$nAsset)
```

```
# finite difference inputs
X$fd$m <- c(10, 10)
X$fd$leftBound <- c(0, 0)
X$fd$kMult <- c(0, 0)
X$fd$density <- c(5, 5)
X$fd$kShift <- c(1, 1)
X$fd$theta <- 0.5
X$fd$maxSmooth <- 2
X$fd$tol <- 1e-7
X$fd$maxIter <- 3

# timestep inputs
X$time$tsType <- 0
X$time$N <- min(X$fd$m) * 4
X$time$dtInit <- 0.1 / 4^log2(min(X$fd$m)/5)
X$time$dNorm <- 5 / 2^log2(min(X$fd$m)/5)
X$time$D <- 0.05

Y <- multiAssetOption(X)

# function check
plotOptionValues(Y, 40)
```

Index