

Package ‘mvQuad’

October 13, 2022

Type Package

Title Methods for Multivariate Quadrature

Version 1.0-6

Date 2016-07-19

Author Constantin Weiser (HHU of Duesseldorf / Germany)

Maintainer Constantin Weiser <weiser@hhu.de>

Description Provides methods to construct multivariate grids, which can be used for multivariate quadrature. This grids can be based on different quadrature rules like Newton-Cotes formulas (trapezoidal-, Simpson's- rule, ...) or Gauss quadrature (Gauss-Hermite, Gauss-Legendre, ...). For the construction of the multidimensional grid the product-rule or the combination- technique can be applied.

URL <https://github.com/weiser/mvQuad/>

License GPL-3

Depends R (>= 3.0)

Imports data.table, statmod, methods

Suggests knitr, rgl, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-20 00:35:33

R topics documented:

mvQuad-package	2
copyNIGrid	3
createNIGrid	3
getNode and getWeights	5
plot (plot.NIGrid)	6

print (print.NIGrid)	7
quadrature	7
QuadRules	8
readRule	9
rescale (rescale.NIGrid)	10
size (size.NIGrid)	11

Index	12
--------------	-----------

mvQuad-package	<i>Methods for multivariate Quadrature (numerical integration)</i>
----------------	--

Description

This package provides methods to construct multivariate grids, which can be used for multivariate quadrature. This grids can be based on different quadrature rules like Newton-Cotes formulas (trapezoidal-, Simpson-rule, ...) or Gauss-Quadrature (Gauss-Hermite, Gauss-Legendre, ...). For the construction of the multidimensional grid the product-rule or the combination-technique can be applied.

Details

Package: mvQuad
 Type: Package
 Version: 1.0-6
 Date: 2016-07-19
 License: GPL-3

Author(s)

Constantin Weiser (HHU of Duesseldorf / Germany) Maintainer: Constantin Weiser <weiser@hhu.de>

References

Philip J. Davis, Philip Rabinowitz (1984): Methods of Numerical Integration
 F. Heiss, V. Winschel (2008): Likelihood approximation by numerical integration on sparse grids, Journal of Econometrics
 H.-J. Bungartz, M. Griebel (2004): Sparse grids, Acta Numerica
 Peter Jaeckel (2005): A note on multivariate Gauss-Hermite quadrature

Examples

```
myGrid <- createNIGrid(dim=2, type="GLe", level=5)
rescale(myGrid, domain=rbind(c(-1,1),c(-1,1)))
```

```

print(myGrid)
plot(myGrid, col="blue")
myFun <- function(x){
  1 - x[,1]^2 * x[,2]^2
}
quadrature(myFun, myGrid)

```

copyNIGrid *copies an NIGrid-object*

Description

copyNIGrid copies an NIGrid-object

Usage

```
copyNIGrid(object1, object2 = NULL)
```

Arguments

object1	original NIGrid-object
object2	destination; if NULL copyNIGrid returns a NIGrid-object otherwise the object2 will be overwritten.

Value

Returns a NIGrid-object or NULL

Examples

```

myGrid <- createNIGrid(dim=2, type="GHe", level=5)
myGrid.copy <- copyNIGrid(myGrid)

```

createNIGrid *creates a grid for numerical integration.*

Description

createNIGrid Creates a grid for multivariate numerical integration. The Grid can be based on different quadrature- and construction-rules.

Usage

```

createNIGrid(dim = NULL, type = NULL, level = NULL,
             ndConstruction = "product", level.trans = NULL)

```

Arguments

<code>dim</code>	number of dimensions
<code>type</code>	quadrature rule (see Details)
<code>level</code>	accuracy level (typically number of grid points for the underlying 1D quadrature rule)
<code>ndConstruction</code>	character vector which denotes the construction rule for multidimensional grids. <code>product</code> for product rule, returns a "full grid" (default) <code>sparse</code> for combination technique, leads to a regular "sparse grid".
<code>level.trans</code>	logical variable denotes either to take the levels as number of grid points (FALSE = default) or to transform in that manner that number of grid points = $2^{(\text{levels}-1)}$ (TRUE). Alternatively <code>level.trans</code> can be a function, which takes (n x d)-matrix and returns a matrix with the same dimensions (see the example; this feature is particularly useful for the 'sparse' construction rule, to account for different importance of the dimensions).

Details

The following quadrature rules are supported (build-in).

`cNC1`, `cNC2`, . . . , `cNC6` closed Newton-Cotes Formula of degree 1-6 (1=trapezoidal-rule; 2=Simpson's-rule; ...), initial interval of integration: [0, 1]

`oNC0`, `oNC1`, . . . , `oNC3` open Newton-Cote Formula of degree 0-3 (0=midpoint-rule; ...), initial interval of integration: [0, 1]

`GLe`, `GKr` Gauss-Legendre and Gauss-Kronrod rule for an initial interval of integration: [0, 1]

`nLe` nested Gauss-Legendre rule for an initial interval of integration: [0, 1] (Knut Petras (2003). Smolyak cubature of given polynomial degree with few nodes for increasing dimension. Numerische Mathematik 93, 729-753)

`GLa` Gauss-Laguerre rule for an initial interval of integration: [0, INF)

`GHe` Gauss-Hermite rule for an initial interval of integration: (-INF, INF)

`nHe` nested Gauss-Hermite rule for an initial interval of integration: (-INF, INF) (A. Genz and B. D. Keister (1996). Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight." Journal of Computational and Applied Mathematics 71, 299-309)

`GHN`, `nHN` (nested) Gauss-Hermite rule as before but weights are multiplied by the standard normal density ($(\hat{w})_i = w_i * \phi(x_i)$).

`Leja` Leja-Points for an initial interval of integration: [0, 1]

The argument `type` and `level` can also be vector-value, different for each dimension (the later only for "product rule"; see examples)

Value

Returns an object of class 'NIGrid'. This object is basically an environment containing nodes and weights and a list of features for this special grid. This grid can be used for numerical integration (via [quadrature](#))

References

- Philip J. Davis, Philip Rabinowitz (1984): Methods of Numerical Integration
F. Heiss, V. Winschel (2008): Likelihood approximation by numerical integration on sparse grids, Journal of Econometrics
H.-J. Bungartz, M. Griebel (2004): Sparse grids, Acta Numerica

See Also

[rescale](#), [quadrature](#), [print](#), [plot](#) and [size](#)

Examples

```
## 1D-Grid --> closed Newton-Cotes Formula of degree 1 (trapezoidal-rule)
myGrid <- createNIGrid(dim=1, type="cNC1", level=10)
print(myGrid)
## 2D-Grid --> nested Gauss-Legendre rule
myGrid <- createNIGrid(dim=2, type=c("GLe","nLe"), level=c(4, 7))
rescale(myGrid, domain = rbind(c(-1,1),c(-1,1)))
plot(myGrid)
print(myGrid)
myFun <- function(x){
  1-x[,1]^2*x[,2]^2
}
quadrature(f = myFun, grid = myGrid)
## level transformation
levelTrans <- function(x){
  tmp <- as.matrix(x)
  tmp[, 2] <- 2*tmp[, 2]
  return(tmp)
}
nw <- createNIGrid(dim=2, type="cNC1", level = 3,
  level.trans = levelTrans, ndConstruction = "sparse")
plot(nw)
```

getNodes and getWeights

get nodes and weights from an NIGrid-object

Description

getNodes and getWeights extract the (potentially rescaled) nodes and weights out of an NIGrid-Object

Usage

```
getNodes(grid)
```

```
getWeights(grid)
```

Arguments

grid object of class NIGrid

Value

Returns the nodes or weights of the given grid

See Also

[createNIGrid](#)

Examples

```
myGrid <- createNIGrid(dim=2, type="cNC1", level=3)
getNode(myGrid)
getWeights(myGrid)
```

plot (plot.NIGrid) *plots an NIGrid-object*

Description

Plots the grid points of an NIGrid-object

Usage

```
## S3 method for class 'NIGrid'
plot(x, plot.dimension = NULL, ...)
```

Arguments

x a grid of type NIGrid
plot.dimension vector of length 1, 2 or 3. with the dimensions to be plotted (see examples)
... arguments passed to the default plot command

Examples

```
myGrid <- createNIGrid(dim=4, type=c("GHe", "cNC1", "GLe", "oNC1"),
                          level=c(3,4,5,6))
plot(myGrid) ## dimension 1-min(3,dim(myGrid)) are plotted
## Free arranged plots
plot(myGrid, plot.dimension=c(4,2,1))
plot(myGrid, plot.dimension=c(1,2))
plot(myGrid, plot.dimension=c(3))
```

`print (print.NIGrid)` *prints characteristic information for an NIGrid-object*

Description

Prints characteristic information for an NIGrid-object

Usage

```
## S3 method for class 'NIGrid'  
print(x, ...)
```

Arguments

`x` a grid of type NIGrid
`...` further arguments passed to or from other methods

Value

Prints the information for an NIGrid-object (i.a. grid size (dimensions, grid points, memory usage), type and support)

Examples

```
myGrid <- createNIGrid(dim=2, type="GHe", level=5)  
print(myGrid)
```

`quadrature` *computes the approximated Integral*

Description

`quadrature` computes the integral for a given function based on an NIGrid-object

Usage

```
quadrature(f, grid = NULL, ...)
```

Arguments

`f` a function which takes the x-values as a (n x d) matrix as a first argument
`grid` a grid of type NIGrid
`...` further arguments for the function `f`

Value

The approximated value of the integral

See Also

[createNIGrid](#), [rescale](#)

Examples

```
myGrid <- createNIGrid(dim=2, type="GLe", level=5)
rescale(myGrid, domain=rbind(c(-1,1),c(-1,1)))
plot(myGrid, col="blue")
myFun <- function(x){
  1 - x[,1]^2 * x[,2]^2
}
quadrature(myFun, myGrid)
```

QuadRules

nodes and weights for 1D - Gauss-Quadrature

Description

This data set stores nodes and weights for Gauss-Quadrature. Syntax:

QuadRules[[*'type'*]][[*'level'*]]

- type="GLe" Gauss-Legendre; interval [0,1]; max-level 45
- type="nLe" nested-type Gauss-Legendre; interval [0,1]; max-level 25
- type="GKr" Gauss-Kronrod; interval [0,1]; max-level 29
- type="GLa" Gauss-Laguerre; interval [0, Inf); max-level 30
- type="GHe" Gauss-Hermite; interval (-Inf, Inf); max-level 45
- type="GHN" Gauss-Hermite (as above, but pre-multiplied weights $\hat{w}_i = w_i * \phi(x_i)$)
- type="nHe" nested-type Gauss-Hermite; interval (-Inf, Inf) max-level 25
- type="nHN" nested-type Gauss-Hermite (as above, but pre-multiplied weights $\hat{w}_i = w_i * \phi(x_i)$)
- type="Leja" Leja-points; interval [0,1]; max-level 141

Format

list of nodes and weights (for organisation see "Syntax" in description section)

Source

- <http://keisan.casio.com/exec/system/1329114617> high precision computing (for G.-rules)
- further information in [createNIGrid](#)

Examples

```
nw <- QuadRules[["GHe"]][[2]]
```

readRule	<i>reads a quadrature-rule from a text file</i>
----------	---

Description

readRule reads a quadrature-rule from a text file

Usage

```
readRule(file = NULL)
```

Arguments

file file name of the text file containing the quadrature rule

Details

The text file containing the quadrature rule has to be formatted in the following way:

The first line have to declare the domain `initial.domain a b`, where `a` and `b` denotes the lower and upper-bound for the integration domain. This can be either a number or `'-Inf'/'Inf'` (for example `initial.domain 0 1` or `initial.domain 0 Inf`)

Every following line contains one single node and weight belonging to one level of the rule (format: `'level' 'node' 'weight'`). This example shows the use for the "midpoint-rule" (levels: 1 - 3).

```
> initial.domain 0 1
> 1 0.5 1
> 2 0.25 0.5
> 2 0.75 0.5
> 3 0.166666666666667 0.333333333333333
> 3 0.5 0.333333333333333
> 3 0.833333333333333 0.333333333333333
```

Value

Returns an object of class `'customRule'`, which can be used for creating a `'NIGrid'` ([createNIGrid](#))

See Also

[createNIGrid](#)

Examples

```
## Not run: myRule <- readRule(file="midpoint_rule.txt")
## Not run: nw <- createNIGrid(d=1, type = myRule.txt, level = 2)
```

```
rescale (rescale.NIGrid)
```

moves, rescales and/or rotates a multidimensional grid.

Description

rescale.NIGrid manipulates a grid for more efficient numerical integration with respect to a given domain (bounded integral) or vector of means and covariance matrix (unbounded integral).

Usage

```
rescale(object, ...)
```

```
## S3 method for class 'NIGrid'
```

```
rescale(object, domain = NULL, m = NULL, C = NULL,
        dec.type = 0, ...)
```

Arguments

object	an initial grid of type NIGrid
...	further arguments passed to or from other methods
domain	a (d x 2)-matrix with the boundaries for each dimension
m	vector of means
C	covariance matrix
dec.type	type of covariance decomposition (<i>Peter Jaeckel (2005)</i>)

Value

This function modifies the "support-attribute" of the grid. The recalculation of the nodes and weights is done when the [getNode](#)s or [getWeights](#) are used.

References

Peter Jaeckel (2005): A note on multivariate Gauss-Hermite quadrature

See Also

[quadrature](#), [createNIGrid](#)

Examples

```
C = matrix(c(2,0.9,0.9,2),2)
m = c(-.5, .3)
par(mfrow=c(3,1))

myGrid <- createNIGrid(dim=2, type="GHe", level=5)
```

```
rescale(myGrid, m=m, C=C, dec.type=0)
plot(myGrid, col="red")

rescale(myGrid, m=m, C=C, dec.type=1)
plot(myGrid, col="green")

rescale(myGrid, m=m, C=C, dec.type=2)
plot(myGrid, col="blue")
```

size (size.NIGrid) *returns the size of an NIGrid-object*

Description

Returns the size of an NIGrid-object

Usage

```
size(object, ...)
```

```
## S3 method for class 'NIGrid'
size(object, ...)
```

```
## S3 method for class 'NIGrid'
dim(x)
```

Arguments

object	a grid of type NIGrid
...	other arguments passed to the specific method
x	object of type NIGrid

Value

Returns the grid size in terms of dimensions, number of grid points and used memory

Examples

```
myGrid <- createNIGrid(dim=2, type="GHe", level=5)
size(myGrid)
dim(myGrid)
```

Index

- * **datasets**
 - QuadRules, 8
- * **math**
 - mvQuad-package, 2
- * **package**
 - mvQuad-package, 2

copyNIGrid, 3
createNIGrid, 3, 6, 8–10

dim.NIGrid(size (size.NIGrid)), 11

getNode, 10
getNode (getNode and getWeights), 5
getNode and getWeights, 5
getWeights, 10
getWeights (getNode and getWeights), 5

mvQuad (mvQuad-package), 2
mvQuad-package, 2

plot, 5
plot (plot.NIGrid), 6
plot.NIGrid(plot (plot.NIGrid)), 6
print, 5
print (print.NIGrid), 7
print.NIGrid(print (print.NIGrid)), 7

quadrature, 4, 5, 7, 10
QuadRules, 8

readRule, 9
rescale, 5, 8
rescale (rescale (rescale.NIGrid)), 10
rescale (rescale.NIGrid), 10
rescale.NIGrid(rescale (rescale.NIGrid)), 10

size, 5
size (size (size.NIGrid)), 11
size (size.NIGrid), 11
size.NIGrid(size (size.NIGrid)), 11