Package ‘volesti’

September 6, 2018

Type Package

License GPL (>= 2)

Title Volume approximation of convex polytopes.

Description Package provides an R interface for VolEsti C++ package. VolEsti computes approximations of volume of polytopes given as a set of points or linear inequalities or Minkowski sum of segments (zonotopes). There are two algorithms for volume approximation as well as algorithms for sampling, rounding and rotating polytopes.

Maintainer Fisikopoulos Vissarion `<vissarion.fysikopoulos@oracle.com>`, Chalkis Apostolos `<tolis.chal@gmail.com>`

Version 1.0.0

Date 2018-09-05

BugReports https://github.com/vissarion/volume_approximation/issues

Imports Rcpp (>= 0.12.17), RcppEigen (>= 0.3.3.4.0), BH (>= 1.66.0-1)

Suggests lpSolveAPI (>= 5.5.2.0-17)

LinkingTo Rcpp (>= 0.12.17), RcppEigen (>= 0.3.3.4.0), BH (>= 1.66.0-1)

RoxygenNote 6.0.1

Author Fisikopoulos Vissarion [cph, cre, aut],
Chalkis Apostolos [cph, ctb, aut]

R topics documented:

`CheBall` *Compute the Chebychev ball of a H-polytope.*

Description

For a H-polytope described by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$, this function computes the largest inscribed ball (Chebychev ball) of that polytope by solving the corresponding linear program. This function needs suggested R-package lpSolveAPI.

Usage

```
CheBall(A, b)
```

Arguments

`A`	The matrix of the H-polytope.
`b`	The m-dimensional vector b that containes the constants of the m facets.

Value

A $(d+1)$-dimensional vector that containes the Chebychev ball. The first d coordinates corresponds to the center and the last one to the radius of the Chebychev ball.

Examples

```
# compute the Chebychev ball of a 2d unit simplex
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
ball_vec = CheBall(A,b)
```

`demoRounding` *Run rounding and rotating tests.*

Description

Choose volume algorithm between CoolingGaussian and SequenceOfBalls and run rounding tests for some skinny cubes. In the first test we apply a random rotation as well before the rounding. We run 10 volume experiments for SequenceOfBalls and 20 for CoolingGaussian and we consider the mean value as the volume approximation.

Usage

```
demoRounding(algo)
```

Arguments

`CG`	The string "CG" to choose CoolingGaussian algorithm
`SOB`	The string "SOB" to choose SequenceOfBalls algorithm

Value

Print the computed volume and print a failure message if the error is larger than the expected.

Examples

```
# run tests for SOB algorithm
demoRounding("SOB")

# run tests for CV algorithm
demoRounding("CG")
```

`demoSampling` *Run some sampling experiments.*

Description

Use uniform or spherical gaussian to sample from some convex H-polytopes, i.e. cubes, simplices, skinny cubes, cross polytopes and birkhoff polytopes. We use the default values, i.e. $walklength = \lfloor 10 + dimension/10 \rfloor$, $N = 100$, Cordinate Directions HnR, $variance = 1$.

Usage

```
demoSampling(distribution)
```

Arguments

`uniform`	The string "uniform" to choose uniform as the target distribution.
`gaussian`	The string "gaussian" to choose spherical gaussian as the target distribution.

Value

Print the computed volumes and the error. If the test fails a message is printed.

Examples

```
# choose uniform distribution
demoSampling("uniform")
# choose spherical gaussian distribution
demoSampling("gaussian")
```

`ExactZonoVol` *Compute the exact volume of a zonotope.*

Description

Given the $m \times d$ matrix that containes the m segments that define the d-dimensional zonotope, this function computes the sum of the determinants of all the $d \times d$ submatrices.

Usage

```
ExactZonoVol(Matrix)
```

Arguments

`Matrix`	The $m \times d$ matrix that containes the segments that define the zonotope.

Value

The exact volume of the zonotope

Examples

```
# compute the exact volume of a 5-dimensional zonotope defined by the Minkowski sum of 10 segments
zonotope = GenZonotope(5, 10)
vol = ExactZonoVol(zonotope)
```

GenCross *Generator function for cross polytopes.*

Description

This function can be used to generate a d-dimensional cross polytope in H or V representation.

Usage

```
GenCross(dimension, repr)
```

Arguments

dimension	The dimension of the cross polytope.
repr	A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A cross polytope in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $2^d \times d$ matrix A and the "vector" containing a 2^d -dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $2d \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional cross polytope in H-representation
PolyList = GenCross(10, 'H')

# generate a 15-dimension cross polytope in V-representation
PolyList = GenCross(15, 'V')
```

GenCube *Generator function for hypercubes.*

Description

This function can be used to generate a d-dimensional Hypercube $[-1, 1]^d$ in H or V representation.

Usage

```
GenCube(dimension, repr)
```

Arguments

`dimension` The dimension of the hypercube

`repr` A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A hypercube in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $2d \times d$ matrix A and the "vector" containing a $2d$-dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $2^d \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional hypercube in H-representation
PolyList = GenCube(10, 'H')

# generate a 15-dimension hypercube in V-representation
PolyList = GenCube(15, 'V')
```

`GenProdSimplex` *Generator function for product of simplices.*

Description

This function can be used to generate a $2d$-dimensional polytope that is defined as the product of two d-dimensional unit simplices in H-representation.

Usage

```
GenProdSimplex(dimension, repr = "H")
```

Arguments

`dimension` The dimension of the simplices.

Value

A polytope defined as the product of two unit simplices in H-representation. The retutn value is a list with two elements: the "matrix" containing a $(2d+1) \times 2d$ matrix A and the "vector" containing a $(2d + 1)$-dimensional vector b, s.t. $Ax \leq b$.

Examples

```
# generate a product of two 5-dimensional simplices.
PolyList = GenProdSimplex(5)
```

GenSimplex *Generator function for simplices.*

Description

This function can be used to generate a d-dimensional unit simplex in H or V representation.

Usage

```
GenSimplex(dimension, repr)
```

Arguments

`dimension`	The dimension of the simplex.
`repr`	A string to declare the representation. It has to be 'H' for H-representation or 'V' for V-representation.

Value

A simplex in H or V-representation. For an H polytope the return value is a list with two elements: the "matrix" containing a $(d+1) \times d$ matrix A and the "vector" containing a $(d+1)$-dimensional vector b, s.t. $Ax \leq b$. When the V-representation is chosen the return value is a $(d+1) \times d$ matrix that containes the vertices row-wise.

Examples

```
# generate a 10-dimensional simplex in H-representation
PolyList = GenSimplex(10, 'H')

# generate a 20-dimensional simplex in V-representation
PolyList = GenSimplex(20, 'V')
```

GenSkinnyCube *Generator function for skinny hypercubes.*

Description

This function can be used to generate a d-dimensional skinny hypercube only in H-representation.

Usage

```
GenSkinnyCube(dimension, repr = "H")
```

Arguments

`dimension`	The dimension of the skinny hypercube.

Value

A d-dimensional skinny hypercube in H-representation. The retutn value is a list with two elements: the "matrix" containing a $2d \times d$ matrix A and the "vector" containing a $2d$-dimensional vector b, s.t. $Ax \leq b$.

Examples

```
# generate a 10-dimensional skinny hypercube.
PolyList = GenSkinnyCube(10)
```

`GenZonotope` *Generator function for zonotopes.*

Description

This function can be used to generate a d-dimensional zonotope described by the Minkowski sum of m segments. We consider the $e_1, \ldots, e_d$ generators and $m - d$ random generators. Then we shift the zonotope in order to contain the origin. The origin is the center of symmetry as well. It might needs rounding before the volume computation.

Usage

```
GenZonotope(dimension, NumGen)
```

Arguments

`dimension`	The dimension of the zonotope.
`NumGen`	The number of segments that generate the zonotope.

Value

A $m \times d$ matrix that containes the m d-dimensional segments.

Examples

```
# generate a 10-dimensional zonotope defined by the Minkowski sum of 20 segments
zonotope = GenZonotope(10, 20)
```

HdemoVolume — *Run some volume approximation experiments for H-polytopes.*

Description

Choose between SequenceOfBalls and CoolingGaussian algorithm to approximate the volume of some cubes, simplices, skinny_cubes, cross polytopes and birkhoff polytopes in H-representation. For each polytope we run 10 volume experiments for SequenceOfBalls and 20 for CoolingGaussian and we consider the mean value as the volume approximation. We demand $error = 0.1$ for the most of them. For all the other parameters we use the default values for both algorithms.

Usage

```
HdemoVolume(algo)
```

Arguments

`CG`	The string "CG" to choose CoolingGaussian algorithm.
`SOB`	The string "SOB" to choose SequenceOfBalls algorithm.

Value

Print the computed volumes and the error. If the test fails a message is printed.

Examples

```
# test SequenceOfBalls
HdemoVolume("SOB")
# test CoolingGausian
HdemoVolume("CG")
```

ineToMatrix — *function to get a ine file and returns a numerical matrix A.*

Description

This function takes an ine file as a string (using read.csv()) and returns a numerical matrix A in ine format for function volume (see $volume$ function examples).

Usage

```
ineToMatrix(P)
```

Arguments

`P`	It is in format, read.cs('path/to/file.ine'). The ine file desrcibes a H-polytope.

Value

The numerical matrix in ine format.

Examples

```
# give the path to birk4.ine
A = ineToMatrix(read.csv('path/to/data/birk4.ine'))
```

---

modifyMat *Takes a numerical matrix in ine format and returns the matrix A and the vector b s.t.: $Ax \leq b$.*

Description

This function can be used to extract from a numerical matrix in ine format (see example), that describes a H-polytope, the $m \times d$ matrix A and the m-dimensional vector b, s.t.: $Ax \leq b$.

Usage

```
modifyMat(A)
```

Arguments

`A`	The numerical matrix in ine format (see example) of the H-polytope.

Value

A list that containes elements "matrix" and "vector", i.e. the numerical $m \times d$ matrix A and the numerical m-dimensional vector b, defining H-polytope P, s.t.: $Ax \leq b$. For V polytopes the element "vector" is useless in practice.

Examples

```
# a 2d unit simplex in H-representation using numerical matrix in ine format
A = matrix(c(3,3,0,0,-1,0,0,0,-1,1,1,1), ncol=3, nrow=4, byrow=TRUE)
list_of_matrix_and_vector = modifyMat(A)
```

rand_rotate *Apply a random rotation to a convex polytope (H-polytope, V-polytope or a zonotope).*

Description

Given a convex H or V polytope or a zonotope as input this function applies a random rotation.

Usage

```
rand_rotate(Inputs)
```

Arguments

`list("argument"=value)`
: A list that includes elements that describe the convex body that is given as input.

`path`
: The path to an ine (H-polytope) or ext (V-polytope, zonotope) file that describes the polytope. If path is given then "matrix" and "vector" inputs are not needed.

`matrix`
: The $m \times d$ matrix A of the H polytope or the $m \times d$ matrix that containes all the m d-dimensional vertices of a V-polytope row-wise or a $m \times d$ matrix that containes all the m d-dimensional segments that define a zonotope row-wise. If the matrix is in ine format, for H-polytopes only (see $volume$ function example), then the "vector" input is not needed.

`vector`
: Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets.

`Vpoly`
: A boolean parameter, has to be true when a V-polytope is given as input. Default value is false.

`Zonotope`
: A boolean parameter, has to be true when a zonotope is given as input. Default value is false.

`verbose`
: Optional. A boolean parameter for printing. Default is false.

Value

A random rotation of the polytope that is given as an input. The output for a H-polytope is a list that containes elements "matrix" and "vector". For a V-polytope the output is a $m \times d$ matrix that containes the m d-dimensional vertices of the V-polytope row-wise. For a zonotope is a $m \times d$ matrix that containes the m d-dimensional segments row-wise.

Examples

```
# rotate a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
listHpoly = rand_rotate(list("matrix"=A, "vector"=b))

# rotate a V-polytope (3d cube)
```

```
V = matrix(c(-1,1,-1,-1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,-1,-1), ncol=3, nrow=8, byrow=TRUE)
matVpoly = rand_rotate(list("matrix"=V, "Vpoly"=TRUE))

# rotate a 5-dimensional zonotope defined by the Minkowski sum of 15 segments
Zono = GenZonotope(5,15)
MatZono = rand_rotate(list("matrix"=Zono, "Zonotope"=TRUE))
```

`round_polytope` *Apply rounding to a convex polytope (H-polytope, V-polytope or a zonotope).*

Description

Given a convex H or V polytope or a zonotope as input this function computes a rounding based on minimum volume enclosing ellipsoid of a pointset.

Usage

```
round_polytope(Inputs)
```

Arguments

`list("argument"=value)`	A list that includes parameters for the rounding.
`path`	The path to an ine (H-polytope) or ext (V-polytope, zonotope) file that describes the polytope. If path is given then "matrix" and "vector" inputs are not needed.
`matrix`	The $m \times d$ matrix A of the H polytope or the $m \times d$ matrix that containes all the m d-dimensional vertices of a V-polytope row-wise or a $m \times d$ matrix that containes all the m d-dimensional segments that define a zonotope row-wise. If the matrix is in ine format, for H-polytopes only (see $volume$ function example), then the "vector" input is not needed.
`vector`	Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets, s.t.: $Ax \leq b$.
`Vpoly`	A boolean parameter, has to be true when a V-polytope is given as input. Default value is false.
`Zonotope`	A boolean parameter, has to be true when a zonotope is given as input. Default value is false.
`walk_length`	Optional. The number of the steps for the random walk, default is $\lfloor 10 + d/10\rfloor$.
`ball_walk`	Optional. Boolean parameter to use ball walk, only for CG algorithm. Default value is false.
`delta`	Optional. The radius for the ball walk.
`coordinate`	Optional. A boolean parameter for the hit-and-run. True for Coordinate Directions HnR, false for Random Directions HnR. Default value is true.
`verbose`	Optional. A boolean parameter for printing. Default is false.

Value

Is a list that containes elements to describe the rounded polytope, i.e. "matrix" and "vector" for H-polytopes and just "matrix" for V-polytopes and zonotopes, containing the verices or segments row-wise. For both representations the list containes element "round_value" which is the determinant of the square matrix of the linear transformation that was applied on the polytope that is given as input.

Examples

```
# rotate a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
listHpoly = round_polytope(list("matrix"=A, "vector"=b))

# rotate a V-polytope (3d cube) using Random Directions HnR
V = matrix(c(-1,1,-1,-1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,-1,-1), ncol=3, nrow=8, byrow=TRUE)
ListVpoly = round_polytope(list("matrix"=V, "Vpoly"=TRUE, "coordinate"=FALSE))

# rotate a 10-dimensional zonotope defined by the Minkowski sum of 20 segments
Zono = GenZonotope(10,20)
ListZono = round_polytope(list("matrix"=Zono, "Zonotope"=TRUE))
```

`sample_points` *Sample points from a convex Polytope (H-polytope, V-polytope or a zonotope).*

Description

Sample N points from a H or a V-polytope or a zonotope with uniform or spherical gaussian -centered in an internal point- target distribution.

Usage

```
sample_points(Inputs)
```

Arguments

`list("argument"=value)`
: A list that includes parameters for the chosen target distribution and the random walk algorithm.

`path`
: The path to an ine (H-polytope) or ext (V-polytope, zonotope) file that describes the polytope. If path is given then "matrix" and "vector" inputs are not needed.

`matrix`
: The $m \times d$ matrix A of the H polytope or the $m \times d$ matrix that containes all the m d-dimensional vertices of a V-polytope row-wise or a $m \times d$ matrix that containes all the m d-dimensional segments that define a zonotope row-wise. If the matrix is in ine format, for H-polytopes only (see $volume$ function example), then the "vector" input is not needed.

`vector`	Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets, s.t.: $Ax \leq b$.
`walk_length`	Optional. The number of the steps for the random walk, default is $\lfloor 10 + d/10 \rfloor$.
`internal_point`	Optional. A d-dimensional vector that containes the coordinates of an internal point of the polytope. If it is not given then for H-polytopes the Chebychev center is computed, for V-polytopes $d+1$ vertices are picked randomly and the Chebychev center of the defined simplex is computed. For a zonotope that is defined by the Minkowski sum of m segments we use the origin.
`gaussian`	Optional. A boolean parameter to sample with gaussian target distribution. Default value is false.
`variance`	Optional. The variance for the spherical gaussian. Default value is 1.
`N`	The number of points that the function is going to sample from the convex polytope. Default value is 100.
`ball_walk`	Optional. Boolean parameter to use ball walk for the sampling. Default value is false.
`delta`	Optional. The radius for the ball walk.
`verbose`	Optional. A boolean parameter for printing. Default is false.
`Vpoly`	A boolean parameter, has to be true when a V-polytope is given as input. Default value is false.
`Zonotope`	A boolean parameter, has to be true when a zonotope is given as input. Default value is false.
`coordinate`	Optional. A boolean parameter for the hit-and-run. True for Coordinate Directions HnR, false for Random Directions HnR. Default value is true.

Value

A $d \times N$ matrix that contains, column-wise, the sampled points from the convex polytope.

Examples

```
# uniform distribution from a 3d cube described by a set of vertices
V = matrix(c(-1,1,-1,-1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,-1,-1), ncol=3, nrow=8, byrow=TRUE)
points = sample_points(list("matrix"=V, "Vpoly"=TRUE, "N"=1000))

# gaussian distribution from a 2d unit simplex in H-representation with variance = 2
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
points = sample_points(list("matrix"=A, "vector"=b, "gaussian"=TRUE, "variance"=2))
```

`VdemoVolume` *Run some volume approximation experiments for V-polytopes.*

Description

Choose between SequenceOfBalls and CoolingGaussian algorithm to approximate the volume of some cubes, simplices and cross polytopes in V-representation. For each polytope we run 10 volume experiments and we consider the mean value as the volume approximation. For SOB algorithm we demand $error = 0.1$ and for CG algorithm we demand $error = 0.2$.

Usage

```
VdemoVolume(algo)
```

Arguments

`CG`	The string "CG" to choose CoolingGaussian algorithm.
`SOB`	The string "SOB" to choose SequenceOfBalls algorithm.

Value

Print the computed volumes and the error. If the test fails a message is printed.

Examples

```
# test SequenceOfBalls
VdemoVolume("SOB")
# test CoolingGausian
VdemoVolume("CG")
```

`volume` *The main R function for volume approximation of a convex Polytope (H-polytope, V-polytope or a zonotope).*

Description

For the volume approximation can be used two algorithms. Either SequenceOfBalls or CoolingGaussian. A H-polytope with m facets is described by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$. A V-polytope is described as a set of d-dimensional points. A zonotope is desrcibed by the Minkowski sum of d-dimensional segments.

Usage

```
volume(Inputs)
```

Arguments

- `list("argument"=value)` A list that includes parameters for the chosen algorithm.
- `path` The path to an ine (H-polytope) or ext (V-polytope, zonotope) file that describes the polytope. If path is given then "matrix" and "vector" inputs are not needed.
- `matrix` The $m \times d$ matrix A of the H polytope or the $m \times d$ matrix that containes all the m d-dimensional vertices of a V-polytope row-wise or a $m \times d$ matrix that containes all the m d-dimensional segments that define a zonotope row-wise. If the matrix is in ine format, for H-polytopes only (see examples), then the "vector" input is not needed.
- `vector` Only for H-polytopes. The m-dimensional vector b that containes the constants of the m facets s.t.: $Ax \leq b$.
- `walk_length` Optional. The number of the steps for the random walk, default is $\lfloor 10 + d/10 \rfloor$.
- `error` Optional. Declare the goal for the approximation error. Default is 1 for SequenceOfBalls and 0.2 for CoolingGaussian.
- `InnerVec` Optional. A $d+1$ vector that containes an inner ball. The first d coordinates corresponds to the center and the last one to the radius of the ball. If it is not given then for H-polytopes the Chebychev ball is computed, for V-polytopes $d+1$ vertices are picked randomly and the Chebychev ball of the defined simplex is computed. For a zonotope that is defined as the Minkowski sum of m segments we compute the maximal r s.t.: $re_i \in Z$ for all $i = 1, \dots, m$.
- `CG` Optional. A boolean parameter to use CoolingGaussian algorithm. Default value is false.
- `win_len` Optional. The size of the window for the ratios' approximation in CG algorithm. Default value is $4\ dimension^2 + 500$.
- `C` Optional. A constant for the lower bound of $variance/mean^2$ in schedule annealing of CG algorithm.
- `N` optional. The number of points we sample in each step of schedule annealing in CG algorithm. Default value is $500C + dimension^2/2$.
- `ratio` Optional. Parameter of schedule annealing of CG algorithm, larger ratio means larger steps in schedule annealing. Default value is $1 - 1/dimension$.
- `frac` Optional. The fraction of the total error to spend in the first gaussian in CG algorithm. Default value is 0.1.
- `ball_walk` Optional. Boolean parameter to use ball walk. Default value is false.
- `delta` Optional. The radius for the ball walk.
- `verbose` Optional. A boolean parameter for printing. Default is false.
- `Vpoly` A boolean parameter, has to be true when a V-polytope is given as input. Default value is false.
- `Zonotope` A boolean parameter, has to be true when a zonotope is given as input. Default value is false.
- `coordinate` Optional. A boolean parameter for the hit-and-run. True for Coordinate Directions HnR, false for Random Directions HnR. Default value is true.
- `rounding` Optional. A boolean parameter to activate the rounding option. Default value is false.

Value

The approximation of the volume of a convex H or V polytope.

References

I.Z.Emiris and V. Fisikopoulos, "Practical polytope volume approximation," ACM Trans. Math. Soft., *2014.*,

B. Cousins and S. Vempala, "A practical volume algorithm," Springer-Verlag Berlin Heidelberg and The Mathematical Programming Society, *2015.*

Examples

```
# calling volesti algorithm for a H-polytope (2d unit simplex)
A = matrix(c(-1,0,0,-1,1,1), ncol=2, nrow=3, byrow=TRUE)
b = c(0,0,1)
vol = volume(list("matrix"=A, "vector"=b))

# calling CV algorithm for a V-polytope (3d cube)
V = matrix(c(-1,1,-1,-1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,-1,-1), ncol=3, nrow=8, byrow=TRUE)
vol = volume(list("matrix"=V, "CG"=TRUE, "Vpoly"=TRUE))

# a 2d unit simplex in H-representation using ine format matrix, calling volesti algorithm
A = matrix(c(3,3,0,0,-1,0,0,0,-1,1,1,1), ncol=3, nrow=4, byrow=TRUE)
vol = volume(list("matrix"=A))

# calling Gaussian-Cooling algorithm for a 5-dimensional zonotope defined as the Minkowski sum of 10 segments
zonotope = GenZonotope(5, 10)
vol = volume(list("matrix"=zonotope, "Zonotope"=TRUE, "rounding"=TRUE, "CG"=TRUE))
```

`ZdemoVolume` *Run some volume approximation experiments for zonotopes.*

Description

Run SequenceOfBalls or CoolingGaussian algorithm to approximate the volume of some zonotopes. In each test we use GenZonotope() function to generate a random zonotope and then we apply rounding before the volume approximation. For each polytope we run 10 volume experiments and we consider the mean value as the volume approximation. For SOB algorithm we demand $error = 0.1$ and for CG algorithm we demand $error = 0.2$.

Usage

```
ZdemoVolume(algo)
```

Arguments

`CG`	The string "CG" to choose CoolingGaussian algorithm.
`SOB`	The string "SOB" to choose SequenceOfBalls algorithm.

Value

Print the computed volumes and the error. If the test fails a message is printed.

Examples

```
# test SequenceOfBalls
ZdemoVolume("SOB")
# test CoolingGausian
ZdemoVolume("CG")
```

Index