Package ‘volesti’

February 24, 2019

Type Package

License LGPL-3

Title Volume Approximation and Sampling 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.

Version 0.0.0

Date 2019-02-24

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

Depends Rcpp (>= 0.12.17)

Imports methods

LinkingTo Rcpp, RcppEigen, BH

Suggests testthat

Encoding UTF-8

RoxygenNote 6.1.1

R topics documented:

`copula1` *Construct a copula using uniform sampling from the unit simplex*

Description

Given two families of parallel hyperplanes intersecting the canonical simplex, this function uniformly samples from the canonical simplex and construct an approximation of the bivariate probability distribution, called copula.

Usage

```
copula1(h1, h2, numSlices, N)
```

Arguments

`h1`	A d-dimensional vector that describes the direction of the first family of parallel hyperplanes.
`h2`	A d-dimensional vector that describes the direction of the second family of parallel hyperplanes.
`numSlices`	The number of the slices for the copula. Default value is 100.
`N`	The number of points to sample. Default value is $4 \cdot 10^6$.

Value

A $numSlices \times numSlices$ numerical matrix that corresponds to a copula.

References

L. Cales, A. Chalkis, I.Z. Emiris, V. Fisikopoulos, "Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises," Proc. of Symposium on Computational Geometry, Budapest, Hungary, *2018.*

Examples

```
# compute a copula for two random families of parallel hyperplanes
h1 = runif(n = 10, min = 1, max = 1000)
h1 = h1 / 1000
h2=runif(n = 10, min = 1, max = 1000)
h2 = h2 / 1000
cop = copula1(h1=h1, h2=h2, numSlices = 10, N = 100000)
```

---

copula2 *Construct a copula using uniform sampling from the unit simplex*

Description

Given a family of parallel hyperplanes and a family of concentric ellispoids centered at the origin intersecting the canonical simplex, this function uniformly samples from the canonical simplex and construct an approximation of the bivariate probability distribution, called copula.

Usage

```
copula2(h, E, numSlices, N)
```

Arguments

`h`	A d-dimensional vector that describes the direction of the first family of parallel hyperplanes.
`E`	The $d \times d$ symmetric positive semidefine matrix that describes the family of concentric ellipsoids centered at the origin.
`numSlices`	The number of the slices for the copula. Default value is 100.
`N`	The number of points to sample. Default value is $4 \cdot 10^6$.

Value

A $numSlices \times numSlices$ numerical matrix that corresponds to a copula.

References

L. Cales, A. Chalkis, I.Z. Emiris, V. Fisikopoulos, "Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises," Proc. of Symposium on Computational Geometry, Budapest, Hungary, *2018.*

Examples

```
# compute a copula for a family of parallel hyperplanes and a family of conentric ellipsoids
h = runif(n = 10, min = 1, max = 1000)
h = h / 1000
E = replicate(10, rnorm(20))
E = cov(E)
cop = copula2(h=h, E=E, numSlices=10, N=100000)
```

---

exact_vol — *Compute the exact volume of (a) a zonotope (b) an arbitrary simplex (c) a unit simplex (d) a cross polytope (e) a hypercube*

Description

Given a zonotope (as an object of class Zonotope), this function computes the sum of the absolute values of the determinants of all the $d \times d$ submatrices of the $m \times d$ matrix G that contains row-wise the m d-dimensional segments that define the zonotope. For an arbitrary simplex that is given in V-representation this function computes the absolute value of the determinant formed by the simplex's points assuming it is shifted to the origin. For a d-dimensional unit simplex, hypercube or cross polytope this function computes the exact well known formulas.

Usage

```
exact_vol(P = NULL, body = NULL, Parameters = NULL)
```

Arguments

`P`	A zonotope or a simplex in V-representation.
`body`	A string that declares the type of the body for the exact sampling: a) 'simplex' for the unit simplex, b) 'cross' for the cross polytope, c) 'hypersphere' for the hypersphere, d) 'cube' for the unit cube.
`Parameters`	A list for the parameters of the methods: • dimension An integer that declares the dimension when exact sampling is enabled for a simplex or a hypersphere. • radius The radius of the d-dimensional hypersphere. Default value is 1.

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
Z = GenZonotope(5, 10)
vol = exact_vol(Z)
```

```
## Not run: # compute the exact volume of a 2-d arbitrary simplex
V = matrix(c(2,3,-1,7,0,0),ncol = 2, nrow = 3, byrow = TRUE)
P = Vpolytope$new(V)
vol = exact_vol(P)

## End(Not run)

# compute the exact volume the 10-dimensional cross polytope
vol = exact_vol(body = "cross", Parameters = list("dimension" = 10))
```

`fileToMatrix` *function to get a ine file and returns a numerical matrix A*

Description

This function takes the path for an ine or an ext file and returns the corresponding numerical matrix and vector that are compatible with volesti package's functions.

Usage

```
fileToMatrix(path, zonotope)
```

Arguments

`path`	A string that containes the path to an ine or a ext file. The ine file desrcibes a H-polytope and ext file describes a V-polytope or a zonotope.
`zonotope`	A boolean parameter. It has to be TRUE when the path leads to an .ext file that describes a zonotope.

Value

A polytope class. If the path corresponds to an ine file then the return value represents a H-polytope. If it corresponds to an ext file the return value represents a V-polytope (default choice) or a zonotope if the second argument is TRUE.

Examples

```
# give the path to birk4.ine
path = system.file('extdata', package = 'volesti')
P = fileToMatrix(paste0(path,'/birk4.ine'))
```

GenCross *Generator function for cross polytopes*

Description

This function can be used to generate the 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 polytope class representing a cross polytope in H- or V-representation.

Examples

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

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

GenCube *Generator function for hypercubes*

Description

This function can be used to generate the d-dimensional unit 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 polytope class representing the unit d-dimensional hypercube in H- or V-representation.

Examples

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

# generate a 15-dimension hypercube in V-representation
P = 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)
```

Arguments

`dimension`	The dimension of the simplices.

Value

A polytope class representing the product of the two d-dimensional unit simplices in H-representation.

Examples

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

GenRandHpoly *Generator function for random H-polytopes*

Description

This function can be used to generate a d-dimensional polytope in H-representation with m facets. We pick m random hyperplanes tangent on the d-dimensional unit hypersphere as facets.

Usage

```
GenRandHpoly(dimension, m)
```

Arguments

dimension	The dimension of the convex polytope.
m	The number of the facets.

Value

A polytope class representing a H-polytope.

Examples

```
# generate a 10-dimensional polytope with 50 facets
P = GenRandVpoly(10, 50)
```

GenRandVpoly *Generator function for random V-polytopes*

Description

This function can be used to generate a d-dimensional polytope in V-representation with m vertices. We pick m random points from the boundary of the d-dimensional unit hypersphere as vertices.

Usage

```
GenRandVpoly(dimension, m)
```

Arguments

dimension	The dimension of the convex polytope.
m	The number of the vertices.

Value

A polytope class representing a V-polytope.

Examples

```
# generate a 10-dimensional polytope defined as the convex hull of 25 random vertices
P = GenRandVpoly(10, 25)
```

`GenSimplex` *Generator function for simplices*

Description

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

Usage

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

Arguments

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

Value

A polytope class representing the d-dimensional unit simplex in H- or V-representation.

Examples

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

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

`GenSkinnyCube` *Generator function for skinny hypercubes*

Description

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

Usage

```
GenSkinnyCube(dimension)
```

Arguments

`dimension`	The dimension of the skinny hypercube.

Value

A polytope class representing the d-dimensional skinny hypercube in H-representation.

Examples

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

`GenZonotope` *Generator function for zonotopes*

Description

This function can be used to generate a random d-dimensional zonotope defined by the Minkowski sum of m d-dimensional segments. We consider m random directions in R^d and for each direction we pick a random length in $[(, \sqrt{d}]$ in order to define m segments.

Usage

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

Arguments

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

Value

A polytope class representing a zonotope.

Examples

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

`Hpolytope` *A C++ class to represent H-polytopes.*

Description

A H-polytope is a convex polytope defined by a set of linear inequalities or equivalently a d-dimensional H-polytope with m facets is defined by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$.

Fields

- `A` A $m \times d$ numerical matrix A
- `b` m-dimensional vector b
- `type` An integer that declares the representation of the polytope. For H-representation the default value is 1.
- `dimension` An integer that declares the dimension of the polytope.

`InnerBall` *Compute an inscribed ball of a convex 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) by solving the corresponding linear program. For a V-polytope $d + 1$ vertices, that define a full dimensional simplex, picked at random and the largest inscribed ball of the simplex is computed. For a zonotope P we compute the minimum r s.t.: $re_i \in P$ for all $i = 1, \dots, d$. Then the ball centered at the origin with radius $r/\sqrt{d}$ is an inscribed ball.

Usage

```
InnerBall(P)
```

Arguments

`P`	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.

Value

A $d+1$-dimensional vector that describes the inscribed ball. The first d coordinates corresponds to the center of the ball and the last one to the radius.

Examples

```
# compute the Chebychev ball of the 2d unit simplex
P = GenSimplex(2,'H')
ball_vec = InnerBall(P)

# compute an inscribed ball of the 3-dimensional unit cube in V-representation
P = GenCube(3, 'V')
ball_vec = InnerBall(P)
```

IntVP — *A C++ class to represent the intersection of two V-polytopes.*

Description

An intersection of two V-polytopes, P_1, P_2, is defined by the intersection of the two coresponding convex hulls.

Fields

- `V1`The numerical matrix that contains the vertices of P_1 row-wise.
- `V2`The numerical matrix that contains the vertices of P_2 row-wise.
- `type`An integer that declares the representation of the polytope. For these kinf of polytopes the default value is 4.
- `dimension`An integer that declares the dimension of the polytope.

---

poly_gen — *An internal Rccp function as a polytope generator*

Description

An internal Rccp function as a polytope generator

Usage

```
poly_gen(kind_gen, Vpoly_gen, dim_gen, m_gen)
```

Arguments

`kind_gen`	An integer to declare the type of the polytope.
`Vpoly_gen`	A boolean parameter to declare if the requested polytope has to be in V-representation.
`dim_gen`	An integer to declare the dimension of the requested polytope.
`m_gen`	An integer to declare the number of generators for the requested random zonotope.

Value

A numerical matrix describing the requested polytope

warning

Do not use this function.

`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(P)
```

Arguments

`P`	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.

Value

A random rotation of the polytope that is given as an input. The return class is the same as the input class.

Examples

```
# rotate a H-polytope (2d unit simplex)
P = GenSimplex(2,'H')
P = rand_rotate(P)

# rotate a V-polytope (3d cube)
P = GenCube(3, 'V')
P = rand_rotate(P)

# rotate a 5-dimensional zonotope defined by the Minkowski sum of 15 segments
Z = GenZonotope(3,6)
Z = rand_rotate(Z)
```

Rcpp_Hpolytope *An exposed C++ class to represent H-polytopes.*

Description

A H-polytope is a convex polytope defined by a set of linear inequalities or equivalently a d-dimensional H-polytope with m facets is defined by a $m \times d$ matrix A and a m-dimensional vector b, s.t.: $Ax \leq b$.

Details

A A $m \times d$ numerical matrix A

b m-dimensional vector b

type An integer that declares the representation of the polytope. For H-representation the default value is 1.

dimension An integer that declares the dimension of the polytope.

Rcpp_IntVP *An exposed C++ class to represent the intersection of two V-polytopes.*

Description

An intersection of two V-polytopes, P_1, P_2, is defined by the intersection of the two coresponding convex hulls.

Details

V1 The numerical matrix that contains the vertices of P_1 row-wise.

V2 The numerical matrix that contains the vertices of P_2 row-wise.

type An integer that declares the representation of the polytope. For these kinf of polytopes the default value is 4.

dimension An integer that declares the dimension of the polytope.

`Rcpp_Vpolytope` *An exposed C++ class to represent V-polytopes.*

Description

A V-polytope is defined as the convex hull of m d-dimensional points which corresponds to its vertices.

Details

`V` A $m \times d$ numerical matrix that contains the vertices row-wise

`type` An integer that declares the representation of the polytope. For V-representation the default value is 2.

`dimension` An integer that declares the dimension of the polytope.

`Rcpp_Zonotope` *An exposed C++ class to represent zonotopes.*

Description

A zonotope is a convex polytope defined by the Minkowski sum of m d-dimensional segments.

Details

`G` A $m \times d$ numerical matrix that contains the segments (or generators) row-wise

`type` An integer that declares the representation of the polytope. For zonotopes the default value is 3.

`dimension` An integer that declares the dimension of the polytope.

`rotating` *An internal Rccp function for the random rotation of a convex polytope*

Description

An internal Rccp function for the random rotation of a convex polytope

Usage

```
rotating(P)
```

Arguments

`P` A convex polytope (H-, V-polytope or a zonotope).

Value

A matrix that describes the rotated polytope

warning

Do not use this function.

---

rounding — *Internal rcpp function for the rounding of a convex polytope*

Description

Internal rcpp function for the rounding of a convex polytope

Usage

```
rounding(P, WalkType = NULL, walk_step = NULL, radius = NULL)
```

Arguments

`P`	A convex polytope (H- or V-representation or zonotope).
`WalkType`	Optional. A string that declares the random walk.
`walk_step`	Optional. The number of the steps for the random walk.
`radius`	Optional. The radius for the ball walk.

Value

A numerical matrix that describes the rounded polytope and contains the round value.

warning

Do not use this function.

`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 functionbrings the polytope in well rounded position based on minimum volume enclosing ellipsoid of a pointset.

Usage

```
round_polytope(P, WalkType = NULL, walk_step = NULL, radius = NULL)
```

Arguments

`P`	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.
`WalkType`	Optional. A string that declares the random walk method: a) 'CDHR' for Coordinate Directions Hit-and-Run, b) 'RDHR' for Random Directions Hit-and-Run or c) 'BW' for Ball Walk. The default walk is 'CDHR'.
`walk_step`	Optional. The number of the steps for the random walk. The default value is $\lfloor 10 + d/10 \rfloor$.
`radius`	Optional. The radius for the ball walk.

Value

A list with 2 elements: (a) a polytope of the same class as the input polytope class and (b) the 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)
P = Hpolytope$new(A, b)
listHpoly = round_polytope(P)

# rotate a V-polytope (3d unit cube) using Random Directions HnR with step equal to 50
P = GenCube(3, 'V')
ListVpoly = round_polytope(P, WalkType = 'RDHR', walk_step = 50)

# round a 2-dimensional zonotope defined by 6 generators using ball walk
Z = GenZonotope(2,6)
ListZono = round_polytope(Z, WalkType = 'BW')
```

`sample_points` *Sample points from a convex Polytope (H-polytope, V-polytope or a zonotope) or use direct methods for uniform sampling from the unit or the canonical or an arbitrary d-dimensional simplex and the boundary or the interior of a d-dimensional hypersphere*

Description

Sample N points with uniform or multidimensional spherical gaussian -centered in an internal point-target distribution. The d-dimensional unit simplex is the set of points $\vec{x} \in \mathrm{R}^d$, s.t.: $\sum_i x_i \leq 1$, $x_i \geq 0$. The d-dimensional canonical simplex is the set of points $\vec{x} \in \mathrm{R}^d$, s.t.: $\sum_i x_i = 1$, $x_i \geq 0$.

Usage

```
sample_points(P = NULL, N = NULL, distribution = NULL,
  WalkType = NULL, walk_step = NULL, exact = NULL, body = NULL,
  Parameters = NULL, InnerPoint = NULL)
```

Arguments

`P`	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.
`N`	The number of points that the function is going to sample from the convex polytope. The default value is 100.
`distribution`	Optional. A string that declares the target distribution: a) 'uniform' for the uniform distribution or b) 'gaussian' for the multidimensional spherical distribution. The default target distribution is uniform.
`WalkType`	Optional. A string that declares the random walk method: a) 'CDHR' for Coordinate Directions Hit-and-Run, b) 'RDHR' for Random Directions Hit-and-Run or c) 'BW' for Ball Walk. The default walk is 'CDHR'.
`walk_step`	Optional. The number of the steps for the random walk. The default value is $\lfloor 10 + d/10 \rfloor$, where d implies the dimension of the polytope.
`exact`	A boolean parameter. It should be used for the uniform sampling from the boundary or the interior of a hypersphere centered at the origin or from the unit or the canonical or an arbitrary simplex. The arbitrary simplex has to be given as a V-polytope. For the rest well known convex bodies the dimension has to be declared and the type of body as well as the radius of the hypersphere.
`body`	A string that declares the type of the body for the exact sampling: a) 'unit simplex' for the unit simplex, b) 'canonical simplex' for the canonical simplex, c) 'hypersphere' for the boundary of a hypersphere centered at the origin, d) 'ball' for the interior of a hypersphere centered at the origin.
`Parameters`	A list for the parameters of the methods: • variance The variance of the multidimensional spherical gaussian. The default value is 1.

- dimension An integer that declares the dimension when exact sampling is enabled for a simplex or a hypersphere.
- radius The radius of the d-dimensional hypersphere. Default value is 1.
- BW_rad The radius for the ball walk.

`InnerPoint` A d-dimensional numerical vector that defines a point in the interior of polytope P.

Value

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

References

R.Y. Rubinstein and B. Melamed, "Modern simulation and modeling" Wiley Series in Probability and Statistics, *1998.*

A Smith, Noah and W Tromble, Roy, "Sampling Uniformly from the Unit Simplex," Center for Language and Speech Processing Johns Hopkins University, *2004.*

Art B. Owen, "Monte Carlo theory, methods and examples," Copyright Art Owen, *2009-2013.*

Examples

```
# uniform distribution from the 3d unit cube in V-representation using ball walk
P = GenCube(3, 'V')
points = sample_points(P, WalkType = "BW", walk_step = 5)

# gaussian distribution from the 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)
P = Hpolytope$new(A,b)
points = sample_points(P, distribution = "gaussian", Parameters = list("variance" = 2))

# uniform points from the boundary of a 10-dimensional hypersphere
points = sample_points(exact = TRUE, body = "hypersphere", Parameters = list("dimension" = 10))

# 10000 uniform points from a 2-d arbitrary simplex
V = matrix(c(2,3,-1,7,0,0),ncol = 2, nrow = 3, byrow = TRUE)
P = Vpolytope$new(V)
points = sample_points(P, N = 10000, exact = TRUE)
```

`SliceOfSimplex` *Compute the percentage of the volume of the unit simplex that is contained in the intersection of a half-space and the unit simplex.*

Description

A half-space H is given as a pair of a vector $a \in R^d$ and a scalar $z0 \in R$ s.t.: $a^T x \leq z0$. This function calls the Ali's version of the Varsi formula to compute a frustum of the unit simplex.

Usage

```
SliceOfSimplex(a, z0)
```

Arguments

a	A d-dimensional vector that defines the direction of the hyperplane.
z0	The scalar that defines the half-space.

Value

The percentage of the volume of the unit simplex that is contained in the intersection of a given half-space and the unit simplex.

References

Varsi, Giulio, "The multidimensional content of the frustum of the simplex," Pacific J. Math. 46, no. 1, 303–314, *1973.*

Ali, Mir M., "Content of the frustum of a simplex," Pacific J. Math. 48, no. 2, 313–322, *1973.*

Examples

```
# compute the frustum of H: -x1+x2<=0
a=c(-1,1)
z0=0
frustum = SliceOfSimplex(a, z0)
```

---

volume — *The main 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 defined as the convex hull of m d-dimensional points which correspond to the vertices of P. A zonotope is desrcibed by the Minkowski sum of m d-dimensional segments.

Usage

```
volume(P, walk_step = NULL, error = NULL, InnerBall = NULL,
  Algo = NULL, WalkType = NULL, rounding = NULL, Parameters = NULL)
```

Arguments

`P`	A convex polytope. It is an object from class (a) Hpolytope or (b) Vpolytope or (c) Zonotope.
`walk_step`	Optional. The number of the steps for the random walk. The default value is $\lfloor 10 + d/10 \rfloor$ for SequenceOfBalls and 1 for CoolingGaussian.
`error`	Optional. Declare the upper bound for the approximation error. The default value is 1 for SequenceOfBalls and 0.1 for CoolingGaussian.
`InnerBall`	Optional. A $d+1$ vector that contains 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 by the Minkowski sum of m segments we compute the maximal r s.t.: $re_i \in Z$ for all $i = 1, \dots, d$, then the ball centered at the origin with radius $r/\sqrt{d}$ is an inscribed ball.
`Algo`	Optional. A string that declares which algorithm to use: a) 'SoB' for SequenceOfBalls or b) 'CG' for CoolingGaussian.
`WalkType`	Optional. A string that declares the random walk method: a) 'CDHR' for Coordinate Directions Hit-and-Run, b) 'RDHR' for Random Directions Hit-and-Run or c) 'BW' for Ball Walk. The default walk is 'CDHR'.
`rounding`	Optional. A boolean parameter for rounding. The default value is FALSE.
`Parameters`	Optional. A list for the parameters of the algorithms: • Window The length of the sliding window for CG algorithm. The default value is $500 + 4 dimension^2$. • C A constant for the lower bound of $variance/mean^2$ in schedule annealing of CG algorithm. The default value is 2. • N The number of points we sample in each step of schedule annealing in CG algorithm. The default value is $500C + dimension^2/2$. • ratio Parameter of schedule annealing of CG algorithm, larger ratio means larger steps in schedule annealing. The default value is $1 - 1/dimension$. • frac The fraction of the total error to spend in the first gaussian in CG algorithm. The default value is 0.1. • BW_rad The radius for the ball walk. The default value is $4r/dimension$, where r is the radius of the inscribed ball of the polytope.

Value

The approximation of the volume of a convex 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 SOB algorithm for a H-polytope (2d unit simplex)
P = GenSimplex(2,'H')
vol = volume(P)

# calling CG algorithm for a V-polytope (3d simplex)
P = GenSimplex(2,'V')
vol = volume(P, Algo = "CG")

# calling CG algorithm for a 2-dimensional zonotope defined as the Minkowski sum of 4 segments
Z = GenZonotope(2, 4)
vol = volume(Z, WalkType = "RDHR", walk_step = 5)
```

`Vpolytope` *A C++ class to represent V-polytopes.*

Description

A V-polytope is defined as the convex hull of m d-dimensional points which corresponds to its vertices.

Fields

- `V` A $m \times d$ numerical matrix that contains the vertices row-wise
- `type` An integer that declares the representation of the polytope. For V-representation the default value is 2.
- `dimension` An integer that declares the dimension of the polytope.

`Zonotope` *A C++ class to represent zonotopes.*

Description

A zonotope is a convex polytope defined by the Minkowski sum of m d-dimensional segments.

Fields

- `G`A $m \times d$ numerical matrix that contains the segments (or generators) row-wise
- `type`An integer that declares the representation of the polytope. For zonotopes the default value is 3.
- `dimension`An integer that declares the dimension of the polytope.

Index