Package ‘LatticeDesign’

August 19, 2025

Version 4.0-1

Title Lattice-Based Space-Filling Designs

Description Lattice-based space-filling designs with fill or separation distance properties including interleaved lattice-based minimax distance designs proposed in Xu He (2017) <doi:10.1093/biomet/asx036>, interleaved lattice-based maximin distance designs proposed in Xu He (2018) <doi:10.1093/biomet/asy069>, interleaved lattice-based designs with low fill and high separation distance properties proposed in Xu He (2024) <doi:10.1137/23M156940X>, (sliced) rotated sphere packing designs proposed in Xu He (2017) <doi:10.1080/01621459.2016.1222289> and Xu He (2019) <doi:10.1080/00401706.2018.1458655>, densest packing-based maximum projections designs proposed in Xu He (2020) <doi:10.1093/biomet/asaa057> and Xu He (2018) <doi:10.48550/arXiv.1709.02062>, maximin distance designs for mixed continuous, ordinal, and binary variables proposed in Hui Lan and Xu He (2025) <doi:10.48550/arXiv.2507.23405>, and optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiments proposed in Xu He, Junpeng Gong, and Zhaohui Li (2025) <doi:10.48550/arXiv.2506.04582>.

License LGPL-2.1

Author Xu He [aut, cre],
Hui Lan [aut],
Junpeng Gong [aut],
Zhaohui Li [aut]

Maintainer Xu He <hexu@amss.ac.cn>

Depends R (>= 3.5.0), methods, nloptr

NeedsCompilation yes

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2025-08-19 12:20:17 UTC

Contents

`AdaptiveRSPD` *Sliced rotated sphere packing designs by enlarging a design*

Description

Generates a sliced rotated sphere packing design by enlarging one rotated sphere packing design.

Usage

```
AdaptiveRSPD(p=2,n,w=100)
```

Arguments

`p`	Number of dimensions, must be an integer greater than one.
`n`	Number of points of the small design, must be a positive integer.
`w`	Number of rotation matrices to try.

Details

This function generates a small rotated sphere packing design and the candidate points for enlarging it.

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`candidates`	The candidate points to add.
`generator`	The generator matrix.

`rotation`	The rotation matrix.
`delta`	The value of parameter delta.
`Theta`	The value of parameter Theta.
`l`	The value of parameter l.
`FillDistance`	The fill distance of the design for the nonboundary region.

References

He, Xu (2018). "Sliced rotated sphere packing designs", *Technometrics*, 61(1): 66-76.

Examples

```
AdaptiveRSPD(p=2,n=50,w=100)
```

`CCs` *Continuous choices*

Description

These data sets give the solutions found from the continuous optimization algorithm. Data sets CCs2, CCs3, CCs4, CCs5, CCs6, CCs7, and CCs8 give the choices in 2, 3, 4, 5, 6, 7, and 8 dimensions, respectively.

Usage

```
data(CCs2);
data(CCs3);
data(CCs4);
data(CCs5);
data(CCs6);
data(CCs7);
data(CCs8);
```

Format

Matrices containing the choices.

References

He, Xu (2024). "Efficient Kriging using interleaved lattice-based designs with low fill and high separation distance properties", *SIAM/ASA Journal on Uncertainty Quantification*, 12(4): 1113-1134.

`CoprimeNumbers` *Coprime mumbers*

Description

Find all potive numbers that is coprime with n and lower than n/2.

Usage

```
CoprimeNumbers(n);
```

Arguments

`n`	A positive integer.

Details

This function finds all potive numbers that is coprime with n and lower than n/2.

Value

The value returned from the function gives the vector containing all potive numbers that are coprime with n and lower than n/2.

References

Xu He, Junpeng Gong, and Zhaohui Li (2025) "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment", <arXiv:2506.04582v1>

Examples

```
CoprimeNumbers(20);
```

`CriterionLLHD` *Criterion value of a lattice-based Latin hypercube design*

Description

Computes the criterion value of a lattice-based Latin hypercube design.

Usage

```
WSL(n,v);
WPL(n,v);
WDL(n,v);
bWSL(n,v);
bWFL(n,v);
bWDL(n,v);
WSL2(n,v);
WFL2(n,v);
WDL2(n,v);
```

Arguments

n	Number of points.
v	Generator vector of the lattice-based Latin hypercube design.

Details

These functions compute the criterion value of a lattice-based Latin hypercube design. WSL, WPL, and WDL give the wrap-around reciprocal separation distance, wrap-around projective separation distance, and wrap-around discrepancy, respectively, for lattice-based Latin hypercube designs. bWSL, bWFL, and bWDL give the bivariate wrap-around reciprocal separation distance, bivariate wrap-around fill distance, and bivariate wrap-around discrepancy, respectively, for lattice-based Latin hypercube designs. WSL2, WFL2, and WDL2 give the wrap-around reciprocal separation distance, wrap-around fill distance, and wrap-around discrepancy, respectively, for two-dimensional lattice-based Latin hypercube designs.

Value

The value returned from the function gives the criterion value of the lattice-based Latin hypercube design.

References

Xu He, Junpeng Gong, and Zhaohui Li (2025) "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment", <arXiv:2506.04582v1>

Examples

```
WSL(100,c(3,7,13));
WPL(100,c(3,7,13));
WDL(100,c(3,7,13));
WSL2(100,c(3,7));
WSL2(100,c(3,13));
WSL2(100,c(13,7));
WFL2(100,c(3,7));
WDL2(101,c(3,7));
bWSL(100,c(3,7,13));
bWFL(100,c(3,7,13));
bWDL(101,c(3,7,13));
```

DPMPD *Densest packing-based maximum projection designs*

Description

Generates a densest packing-based maximum projection design.

Usage

```
DPMPD(p,n,rotation="magic",w=100)
```

Arguments

`p`	Number of dimensions, must be an integer greater than one and no higher than eight.
`n`	Number of points, must be an integer greater than one.
`rotation`	Optional, whether to use magic rotation matrices (for p=2,3,4,6,8, recommended) or random rotation matrices.
`w`	Number of rotation matrices to try.

Details

This function generates a densest packing-based maximum projection design in two to eight dimensions. For p=2,4,8 with rotation="magic", the designs are generated following the Biometrika paper "Lattice-based designs possessing quasi-optimal separation distance on all projections". For p=3,6 with rotation="magic", the designs are generated following the arXiv paper "Lattice-based designs with quasi-uniform projections". For other p or rotation!="magic", the designs are generated from random rotations.

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`ProjectiveSeparationDistance`	The projective separation distance of the generated design, from one-dimensional projections to the unprojected design.

References

He, Xu (2020). "Lattice-based designs possessing quasi-optimal separation distance on all projections", *Biometrika*, accepted, DOI:10.1093/biomet/asaa057.

He, Xu (2018). "Lattice-based designs with quasi-uniform projections", arXiv:1709.02062v2.

Examples

```
DPMPD(p=4,n=200,w=100)
```

`GeneratorMatrices` *Generator matrices of standard interleaved lattices, treating dimension permuted lattices as different lattices*

Description

These data sets give the generator matrices of standard interleaved lattices, treating dimension permuted lattices as different lattices. Data sets GeneratorMatrices2, GeneratorMatrices3, GeneratorMatrices4, and GeneratorMatrices5 give the matrices in 2, 3, 4, and 5, dimensions, respectively.

Usage

```
data(GeneratorMatrices2);
data(GeneratorMatrices3);
data(GeneratorMatrices4);
data(GeneratorMatrices5);
```

Format

Matrices containing generator matrices.

References

He, Xu (2019). "Interleaved lattice-based maximin distance designs", *Biometrika*, 106(2): 453-464.

`GMs` *Generator matrices of standard interleaved lattices, treating dimension permuted lattices as the same lattice*

Description

These data sets give the generator matrices of standard interleaved lattices, treating dimension permuted lattices as the same lattice. Data sets GMs2, GMs3, GMs4, GMs5, GMs6, GMs7, and GMs8 give the matrices in 2, 3, 4, 5, 6, 7, and 8 dimensions, respectively.

Usage

```
data(GMs2);
data(GMs3);
data(GMs4);
data(GMs5);
data(GMs6);
data(GMs7);
data(GMs8);
```

Format

Matrices containing generator matrices.

References

He, Xu (2017). "Interleaved lattice-based minimax distance designs", *Biometrika*, 104(3): 713-725.

`InterleavedFillSepD` *Interleaved lattice-based fill and separation distance designs*

Description

Generates an interleaved lattice-based designs with low fill and high separation distance properties.

Usage

```
InterleavedFillSepD(p,n,w=rep(1,p),pfrom=p,a=1/2,nmin=floor(n*.8),
  nmax=ceiling(n*1.2),coefF=-4,coefS=1,msC=0,NL=10,NP=100,NJ=10,NS=100);
```

Arguments

`p`	Number of dimensions.
`n`	Targeted number of points, must be an integer greater than one.
`w`	Optional, weights of the dimensions.
`pfrom`	Optional, number of dimensions designs are generated and supplemented from, no more than p and no more than 8.
`a`	Optional, translation parameter, with a=0 for uniform design and a=1 for pushing the points to the boundary.
`nmin`	Optional, minimal acceptable number of points, no less than n.
`nmax`	Optional, maximal acceptable number of points, no greater than n.
`coefF`	Optional, coefficient of r_F in the criterion.
`coefS`	Optional, coefficient of r_S in the criterion.
`msC`	Optional, maximal allowed sum of project weights for pairs of binary aliased dimensions.
`NL`	Optional, maximum number of lattices to try.
`NP`	Optional, maximum number of dimension permutations to try.
`NJ`	Optional, maximum number of discretization choices for each continuous lattice and s combination.
`NS`	Optional, maximum number of choices to supplement from.

Details

This function generate an interleaved lattice-based designs with low fill and high separation distance properties in p dimensions and around n points, following the algorithm provided in the paper "Efficient Kriging using interleaved lattice-based designs with low fill and high separation distance properties".

Value

The value returned from the function is a matrix containing the generated design. Remark that no qualified design might be found if (a) both nmax and nmin are to close to n, (b) both n and msC are small, or (c) both pfrom and msC are small while p-pfrom is big.

References

He, Xu (2024). "Efficient Kriging using interleaved lattice-based designs with low fill and high separation distance properties", *SIAM/ASA Journal on Uncertainty Quantification*, 12(4): 1113-1134.

Examples

```
InterleavedFillSepD(p=2,n=20);
```

`InterleavedMaximinD` *Interleaved lattice-based maximin distance designs*

Description

Generates an interleaved lattice-based maximin distance design.

Usage

```
InterleavedMaximinD(p,n,weight=rep(1,p));
InterleavedMaximinDAlg1(p,n,weight=rep(1,p));
InterleavedMaximinDAlg2(p,n,weight=rep(1,p));
InterleavedMaximinDAlg3(p,n,weight=rep(1,p));
```

Arguments

`p`	Number of dimensions, must be an integer greater than one.
`n`	Targeted number of points, must be an integer greater than one.
`weight`	Optional, the weights used in the distance measure, higher for more important variable.

Details

This function generates an interleaved lattice-based maximin distance design in p dimensions and at least n points, following the algorithms provided in the paper "Interleaved lattice-based maximin distance designs". Function InterleavedMaximinD uses the recommended algorithm provided in the paper. Functions InterleavedMaximinDAlg1, InterleavedMaximinDAlg2, and InterleavedMaximinDAlg3 use Algorithm 1, 2, and 3, respectively. For InterleavedMaximinDAlg1, p must be no greater than 5. For InterleavedMaximinDAlg3, p must be greater than 8.

Value

The value returned from the function is a list containing the following components:

- `Design` The generated design.
- `SeparationDistance` The separation distance of the generated design.
- `m` The actual number of points of the generated design.
- `DesignTransformed` The generated design that is transformed to the rectangular design space given the weights.
- `weight` The weight used in the distance measure, higher for more important variable.
- `s_vector` The numbers of distinct levels of the generated design.
- `L01` The base design.

References

He, Xu (2019). "Interleaved lattice-based maximin distance designs", *Biometrika*, 106(2): 453-464.

Examples

```
InterleavedMaximinD(p=3,n=10,weight=rep(1,3));
InterleavedMaximinDAlg1(p=3,n=10);
InterleavedMaximinDAlg2(p=6,n=10);
InterleavedMaximinDAlg3(p=9,n=257);
```

InterleavedMaximinDMixVars *Interleaved lattice-based maximin distance designs for mixed continuous, ordinal, and binary variables*

Description

Generates an interleaved lattice-based maximin distance design for mixed continuous, ordinal, and binary variables.

Usage

```
InterleavedMaximinDMixVars(p,n,discrete_dims,ordinal_levels,weight=rep(1,p));
ILMmDMixVarsAlg6(p,n,discrete_dims,ordinal_levels,weight=rep(1,p));
ILMmDMixVarsAlg8(p,n,discrete_dims,ordinal_levels,weight=rep(1,p),N=1,ReturnAll=0);
ILMmDMixVarsAlg9(p,n,discrete_dims,ordinal_levels,weight=rep(1,p),pfrom=8);
```

Arguments

`p`	Number of dimensions, must be an integer greater than one.
`n`	Targeted number of points, must be an integer greater than one.
`discrete_dims`	Index of discrete variables in 1:p.
`ordinal_levels`	Allowable levels of ordinal variables by a matrix.
`weight`	Optional, the weights used in the distance measure, higher for more important variable.
`N`	Number of results saved.
`ReturnAll`	Whether or not outputing all designs with the highest separation distance, regardless of their sample size.
`pfrom`	Number of dimensions to supplement from.

Details

These functions generate an interleaved lattice-based maximin distance design for mixed continuous, ordinal, and binary variables in p dimensions and at most n points, following the algorithms provided in the paper "Maximin distance designs for mixed continuous, ordinal, and binary variables". Function InterleavedMaximinDMixVars uses the recommended algorithm provided in the paper. Functions ILMmDMixVarsAlg6, ILMmDMixVarsAlg8, and ILMmDMixVarsAlg9 use Algorithms 6, 8, and 9, respectively. For ILMmDMixVarsAlg6, p must be no greater than 5. For ILMmDMixVarsAlg8, p must be no greater than 8.

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`SeparationDistance`	The separation distance of the generated design.
`m`	The actual number of points of the generated design.
`DesignTransformed`	The generated design that is transformed to the rectangular design space given the weights.
`weight`	The weight used in the distance measure, higher for more important variable.
`s_vector`	The numbers of distinct levels of the generated design.
`L01`	The base design.

References

Lan, Hui and He, Xu (2025). "maximin distance designs for mixed continuous, ordinal, and binary variables", arXiv:2507.23405v1.

Examples

```
ordinal_levels_inpaper <- list(
  Q1 = c(0, 1),
  Q2 = c(0, 0.5, 1),
  Q3 = c(0, 0.1, 0.3, 0.6, 1),
  Q4 = c(0, 0.25, 0.5, 0.75, 1),
  Q5 = c(0, 0.2, 0.3, 0.5, 0.7, 1),
  Q6 = c(0, 0.2, 0.4, 0.6, 0.8, 1),
  Q7 = c(0, 0.1, 0.3, 0.5, 0.7, 0.9, 1),
  Q8 = c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1)
)

discrete_dims <- c(2,4)
ordinal_levels <- list(
  v1 = ordinal_levels_inpaper$Q6,
  v2 = ordinal_levels_inpaper$Q7
)
res <- InterleavedMaximinDMixVars(p=4,n=50,discrete_dims,ordinal_levels,weight=(3/4)^(0:3));
res$Design

discrete_dims <- c()
ordinal_levels <- c()
res <- InterleavedMaximinDMixVars(p=3,n=50,discrete_dims,ordinal_levels,weight=rep(1,3));
res$Design

discrete_dims <- c(1,2,3)
ordinal_levels <- list(
  v1 = ordinal_levels_inpaper$Q8,
  v2 = ordinal_levels_inpaper$Q3,
  v3 = ordinal_levels_inpaper$Q8
)
res <- ILMmDMixVarsAlg6(p=3,n=30,discrete_dims,ordinal_levels);
res$Design

discrete_dims <- c(3,4,5)
ordinal_levels <- list(
  v1 = ordinal_levels_inpaper$Q1,
  v2 = ordinal_levels_inpaper$Q5,
  v3 = ordinal_levels_inpaper$Q8
)
res <- ILMmDMixVarsAlg8(p=5,n=40,discrete_dims,ordinal_levels);
res$Design

discrete_dims <- c(4:10)
ordinal_levels <- list(
  v1 = ordinal_levels_inpaper$Q1,
  v2 = ordinal_levels_inpaper$Q1,
```

```
  v3 = ordinal_levels_inpaper$Q1,
  v4 = ordinal_levels_inpaper$Q2,
  v5 = ordinal_levels_inpaper$Q3,
  v6 = ordinal_levels_inpaper$Q4,
  v7 = ordinal_levels_inpaper$Q5
)
res <- ILMmDMixVarsAlg9(p=10,n=60,discrete_dims,ordinal_levels,pfrom=5);
res$Design
```

`InterleavedMinimaxD` *Interleaved lattice-based minimax distance designs*

Description

Generates an interleaved lattice-based minimax distance design.

Usage

```
InterleavedMinimaxD(p,n,maxdissimilarity=2*p);
```

Arguments

`p`	Number of dimensions, must be an integer between 2 and 8.
`n`	Targeted number of points, must be an integer greater than one.
`maxdissimilarity`	Optional, the maximum dissimilarity allowed for the number of levels.

Details

These functions generate an interleaved lattice-based minimax distance design in p dimensions and at most n points, following the algorithm provided in the paper "Interleaved lattice-based minimax distance designs".

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`TargetFillDistance`	The target fill distance, an estimate of the fill distance.
`ActualSize`	The actual number of points of the generated design.
`s_vector`	The numbers of distinct levels of the generated design.
`L01`	The base design.

References

He, Xu (2017). "Interleaved lattice-based minimax distance designs", *Biometrika*, 104(3): 713-725.

Examples

```
InterleavedMinimaxD(p=2,n=20);
```

LLHD *Optimal generator vector for Lattice-based Latin hypercube designs*

Description

Generates the optimal generator vector for lattice-based Latin hypercube designs.

Usage

```
LLHD(n,d,criterion="WS",T=1000,nstart=max(c(floor(T/(length(CoprimeNumbers(n))*d*5)),1)));
```

Arguments

n	Number of points, must be an integer greater than one.
d	Number of dimensions, must be an integer greater than one.
criterion	Optional, the criterion used in optimization, which can be "WS", "WP", "WD", "bWS", "bWF", or "bWD".
T	Optional, number of iterations.
nstart	Optional, number of random starts.

Details

This function generates the optimal generator vector for lattice-based Latin hypercube designs, following the algorithms provided in the paper "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment".

Value

The value returned from the function gives a generator vector for lattice-based Latin hypercube designs.

References

Xu He, Junpeng Gong, and Zhaohui Li (2025) "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment", <arXiv:2506.04582v1>

Examples

```
set.seed(85)
v <- LLHD(n=20,d=3,criterion="WS")
design <- LLHDpoints(n=20,v=v,delta=sample(x=1:20,size=3,replace=TRUE))
design
```

LLHDpoints *Points of lattice-based Latin hypercube designs*

Description

Generates the points of a lattice-based Latin hypercube design.

Usage

```
LLHDpoints(n,v,delta);
```

Arguments

`n`	Number of points, must be an integer greater than one.
`v`	Generator vector.
`delta`	Translation parameter.

Details

This function gives the points of a lattice-based Latin hypercube design, following the formulas provided in the paper "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment".

Value

The value returned from the function gives the design matrix for the lattice-based Latin hypercube design.

References

Xu He, Junpeng Gong, and Zhaohui Li (2025) "optimized and regularly repeated lattice-based Latin hypercube designs for large-scale computer experiment", <arXiv:2506.04582v1>

Examples

```
set.seed(85)
v <- LLHD(n=20,d=3,criterion="WS")
design <- LLHDpoints(n=20,v=v,delta=sample(x=1:20,size=3,replace=TRUE))
design
```

LRS *Vertexes of a polytope giving halfspace definition*

Description

Computes the radius, widths, and vertexes of a polytope giving halfspace definition. The program is a R shell of LRS (v.5.1a with lrsmp.h), a reverse search vertex enumeration program/CH package in C which is developed by David Avis <http://cgm.cs.mcgill.ca/~avis/C/lrs.html>. Consider the problem of Ax<=b, where A is an n*p matrix, x is a p-vector, and b is an n-vector. Please make sure that the solution of x is nonempty and bounded. Then the nonequalities give the halfspace definition of a polytope. Also make sure that A and b are rational numbers.

Usage

```
LRS(numerator,denominator);
```

Arguments

`numerator`	The numerators of cbind(b,A), an n*(p+1) matrix of integer numbers.
`denominator`	The denominators of cbind(b,A), an n*(p+1) matrix of integer numbers.

Details

This function computes the radius, widths, and vertexes of a polytope giving halfspace definition. It is used in constructing interleaved lattice-based minimax distance designs. Currently only tested when the maximum values of numerators and denominators are below 2^20. If the nonequalities are not defined by rational numbers, round-up to small rational numbers is needed before calling the function. The computation is slow for large p but very fast for slow p. Avoid redundant nonequalities may accelerate the calculation.

Value

The value returned from the function is a list containing the following components:

`Radius`	The maximum L2 distance of vertexes to the origin.
`MaxValue`	The maximum k-dimensional value of the vertexes, for k from 1 to p.
`Vertexes`	The vertexes of the polytope.

References

Avis, David. LRS, http://cgm.cs.mcgill.ca/~avis/C/lrs.html.

He, Xu (2017). "Interleaved lattice-based minimax distance designs", *Biometrika*, 104(3): 713-725.

See Also

`InterleavedMinimaxD`.

Examples

```
num = matrix(0,5,3)
den = matrix(1,5,3)
num[1,2] = -1;  den[1,2] = 2;
num[1,1] = 1;   den[1,1] = 8;
num[2,3] = -1;
num[2,1] = 1;   den[2,1] = 2;
num[3,2] = -1;  den[3,2] = 4;
num[3,3] = -1;  den[3,3] = 2;
num[3,1] = 5;   den[3,1] = 32;
num[4,2] = 1;
num[4,1] = 0;
num[5,3] = 1;
num[5,1] = 0;
LRS(num,den)
```

ProjSepD — *Projective separation distance of a design*

Description

Computes the projective separation distance of a design.

Usage

```
ProjSepD(design);
```

Arguments

design	The experimental design, must be a matrix whose rows indicate experimental runs.

Details

This function computes the squared projective separation distance of a design.

Value

The value returned from the function gives the squared one-dimensional, two-dimensional, ..., (p-1)-dimensional projective separation distances, and the unprojected separation distance, where p is the number of dimensions of the design.

References

He, Xu (2020). "Lattice-based designs possessing quasi-optimal separation distance on all projections", *Biometrika*, accepted, DOI:10.1093/biomet/asaa057.

Examples

```
design = rbind(1:3,c(41,1.2,1.3),c(5.4,5.48,5.7),c(4.3,2.3,2));
ProjSepD(design);
```

RSPD *Rotated sphere packing designs*

Description

Generates a rotated sphere packing design.

Usage

```
RSPD(p=2,n,rotation="magic",w=100)
```

Arguments

`p`	Number of dimensions, must be an integer greater than one.
`n`	Number of points, must be a positive integer.
`rotation`	Optional, whether to use the magic rotation matrix (for p=2, recommended) or random rotation matrices.
`w`	Number of rotation matrices to try, fixed to 1 when p=2 and rotation="magic".

Details

This function generates a rotated sphere packing design.

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`generator`	The generator matrix.
`rotation`	The rotation matrix.
`delta`	The value of parameter delta.
`Theta`	The value of parameter Theta.
`l`	The value of parameter l.
`FillDistance`	The fill distance of the design for the nonboundary region.

References

He, Xu (2017). "Rotated sphere packing designs", *Journal of the American Statistical Association*, 112(520): 1612-1622.

Examples

```
RSPD(p=2,n=50,rotation="magic",w=100)
```

SlicedRSPD *Sliced rotated sphere packing designs by partitioning a design*

Description

Generates a sliced rotated sphere packing design by partitioning one rotated sphere packing design.

Usage

```
SlicedRSPD(p=2,n,rotation="magic",w=100)
```

Arguments

`p`	Number of dimensions, must be an integer greater than one.
`n`	Number of points, must be a positive integer.
`rotation`	Optional, whether to use magic rotation matrices (for p=2, recommended) or random rotation matrices.
`w`	Number of rotation matrices to try.

Details

This function generates a rotated sphere packing design and the slice indexes of points.

Value

The value returned from the function is a list containing the following components:

`Design`	The generated design.
`slices`	The slice indexes of design points.
`generator`	The generator matrix.
`rotation`	The rotation matrix.
`delta`	The value of parameter delta.
`Theta`	The value of parameter Theta.
`l`	The value of parameter l.
`FillDistance`	The fill distance of the design for the nonboundary region.

References

He, Xu (2019). "Sliced rotated sphere packing designs", *Technometrics*, 61(1): 66-76.

Examples

```
SlicedRSPD(p=2,n=50,rotation="magic",w=100)
```

Index