Space-Filling Designs for Computer Experiments

Holger H. Hoos

based on Chapter 5 of T.J. Santner *et al.*:
The Design and Analysis of Computer Experiments, Springer, 2003.

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)
- Space filling designs and basic methods for generating them, in particular, Latin hypercube designs (5.2)

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)
- Space filling designs and basic methods for generating them, in particular, Latin hypercube designs (5.2)
- Briefly address weaknesses of simple LHDs and some basic approaches for overcoming them (p.130*f.*, 5.2.4)

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)
- Space filling designs and basic methods for generating them, in particular, Latin hypercube designs (5.2)
- Briefly address weaknesses of simple LHDs and some basic approaches for overcoming them (p.130*f.*, 5.2.4)
- Discuss measures for spread and distance-based designs (5.3)

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)
- Space filling designs and basic methods for generating them, in particular, Latin hypercube designs (5.2)
- Briefly address weaknesses of simple LHDs and some basic approaches for overcoming them (p.130*f.*, 5.2.4)
- Discuss measures for spread and distance-based designs (5.3)
- Discuss uniform designs (5.4)

Goals

- Review basic principles of experimental design (= input selection) and their applicability to 'computer experiments' (5.1.1, 5.1.2)
- Space filling designs and basic methods for generating them, in particular, Latin hypercube designs (5.2)
- Briefly address weaknesses of simple LHDs and some basic approaches for overcoming them (p.130*f.*, 5.2.4)
- Discuss measures for spread and distance-based designs (5.3)
- Discuss uniform designs (5.4)
- Briefly discuss designs satisfying combinations of criteria (5.5)

Introduction

- Experimental design = selection of inputs at which to compute output of computer experiment to achieve specific goals
- Chapters 5 and 6 of DACE covers different methods for doing this

Introduction

- Experimental design = selection of inputs at which to compute output of computer experiment to achieve specific goals
- Chapters 5 and 6 of DACE covers different methods for doing this
- Terminology:
  - *experimental region:* set of (combinations of) input values for which we wish to study or model response
    point in experimental region: specific set of input values
  - *experimental design:* set of points in experimental region for which we compute the response

Some Basic Principles of Experimental Design

- **Goal:** study how response varies as inputs are changed.

Some Basic Principles of Experimental Design

- **Goal:** study how response varies as inputs are changed.
- In physical experiments (or any other scenario with uncontrolled factors) this is is complicated by
  - noise (unsystematic effect of uncontrolled factors)
  - bias (systematic effect of uncontrolled factors)

Some Basic Principles of Experimental Design

- **Goal:** study how response varies as inputs are changed.
- In physical experiments (or any other scenario with uncontrolled factors) this is is complicated by
  - noise (unsystematic effect of uncontrolled factors)
  - bias (systematic effect of uncontrolled factors)
- Classical experimental design uses
  - *replication* and *blocking* to control for noise
  - *randomisation* to control for bias

Some Basic Principles of Experimental Design

- **Goal:** study how response varies as inputs are changed.
- In physical experiments (or any other scenario with uncontrolled factors) this is is complicated by
  - noise (unsystematic effect of uncontrolled factors)
  - bias (systematic effect of uncontrolled factors)
- Classical experimental design uses
  - *replication* and *blocking* to control for noise
  - *randomisation* to control for bias
- In (deterministic) 'computer experiments', noise and bias don't occur, so replication, blocking and randomisation are not needed.

Additional complications can arise from:

- Correlated inputs (*collinearity*)

Additional complications can arise from:

- Correlated inputs (*collinearity*)
- Incorrect assumptions in the statistic model of the relation between inputs and response (*model bias)*

Additional complications can arise from:

- Correlated inputs (*collinearity*)
- Incorrect assumptions in the statistic model of the relation between inputs and response (*model bias)*
- Experimental design methods are used to address these problems:
  - *orthogonal design*: use of uncorrelated input values makes it possible to independently assess effects of individual inputs on response (see also *factorial designs*)
  - designs for model bias + use of diagnostics (*e.g.*, scatter plots, quantile plots) can protect against certain types of bias

Optimal designs:

- formulate purpose of experiment in terms of optimising an objective f
- select design such that design (*i.e.*, set of points from experimental region) optimises f

Optimal designs:

- formulate purpose of experiment in terms of optimising an objective f
- select design such that design (*i.e.*, set of points from experimental region) optimises f

Example:

- Fit straight line to given data
- Goal: select design to give most precise (min variance) estimate of slope

Some common objectives for linear models:

- minimise generalised variance of least squares estimates of model parameters (determinant of covariance matrix)
  $\rightsquigarrow$ D-optimal designs
- minimise average variance (trace of covariance matrix)
  $\rightsquigarrow$ A-optimal designs
- minimise average of predicted response over experimental region
  $\rightsquigarrow$ I-optimal designs

Note:

- Many experiments have multiple goals and it is unclear how to formulate an optimisation objective.
- Even if an optimisation objective has been formulated it, finding optimal designs can be difficult.
- Chapter 6 will look further into optimal design; as it turns out, one has to resort to heuristic optimisation methods for practical implementations.

‘Computer experiments’ are deterministic, therefore:

- the only source of error is model bias
  Note: In many cases there will be a trade-off between model accuracy and model complexity. At least in cases where one experimental goal is to gain a better understanding of the behaviour of the algorithm, e.g., for the purpose of improving it, highly complex models may be undesirable.

'Computer experiments' are deterministic, therefore:

- the only source of error is model bias

  Note: In many cases there will be a trade-off between model accuracy and model complexity. At least in cases where one experimental goal is to gain a better understanding of the behaviour of the algorithm, e.g., for the purpose of improving it, highly complex models may be undesirable.

- Designs should not take more than one observation for any set of inputs. (If the code and the execution environment do not change.)

'Computer experiments' are deterministic, therefore:

- the only source of error is model bias

  Note: In many cases there will be a trade-off between model accuracy and model complexity. At least in cases where one experimental goal is to gain a better understanding of the behaviour of the algorithm, e.g., for the purpose of improving it, highly complex models may be undesirable.

- Designs should not take more than one observation for any set of inputs. (If the code and the execution environment do not change.)

- Designs should allow one to fit a variety of models.

'Computer experiments' are deterministic, therefore:

- the only source of error is model bias
  Note: In many cases there will be a trade-off between model accuracy and model complexity. At least in cases where one experimental goal is to gain a better understanding of the behaviour of the algorithm, e.g., for the purpose of improving it, highly complex models may be undesirable.
- Designs should not take more than one observation for any set of inputs. (If the code and the execution environment do not change.)
- Designs should allow one to fit a variety of models.
- Designs should provide information about all portions of experimental region. (If there is no prior knowledge / assumptions about true relationship between inputs and response.)

As a corrolary of the last principle, one should use *space-filling designs*, *i.e.*, designs that spread points evenly throughout experimental region.

As a corrolary of the last principle, one should use *space-filling designs*, *i.e.*, designs that spread points evenly throughout experimental region.

Another reason for the use of space-filling designs:

- predictors for response are often based on interpolators (*e.g.*, best linear unbiased predictors from Ch.3)
- prediction error at any point is relative to its distance from clostest design point
- uneven designs can yield predictors that are very inaccurate in sparsely observed parts of experimental region

Simple Designs

Select points using ...

- **regular grid** over experimental region

Simple Designs

Select points using ...

- **regular grid** over experimental region
- *simple random sampling*

Simple Designs

Select points using ...

- **regular grid** over experimental region
- *simple random sampling*
  for small samples in high-dimensional regions often exhibits clustering and poorly covered areas

Simple Designs

Select points using ...

- **regular grid** over experimental region
- *simple random sampling*
  for small samples in high-dimensional regions often exhibits clustering and poorly covered areas
- *stratified random sampling:*
  - divide region into n strata (spread evenly), sample one point
  - randomy select one point from each stratum

Latin Hypercube Designs (LHDs)

Motivation:

- if we expect that output depends only on few of the inputs (*factor sparsity*), points should be evenly spaced when projecting onto experimental region onto these factors

Latin Hypercube Designs (LHDs)

Motivation:

- if we expect that output depends only on few of the inputs (*factor sparsity*), points should be evenly spaced when projecting onto experimental region onto these factors
- if we assume (approximately) additive model, we also want a design whose points are projected evenly over the values of individual inputs

Latin Hypercube Designs (LHDs)

Motivation:

- if we expect that output depends only on few of the inputs (*factor sparsity*), points should be evenly spaced when projecting onto experimental region onto these factors
- if we assume (approximately) additive model, we also want a design whose points are projected evenly over the values of individual inputs
- it can be shown that (at least under some assumptions), LHDs are better than (equally sized) designs obtained from simple random sampling

How to construct an LHD with n points for two continuous inputs:

1. partition experimental region into a square with n^2 cells (n along each dimension)

How to construct an LHD with n points for two continuous inputs:

1. partition experimental region into a square with n^2 cells (n along each dimension)
2. labels the cells with integers from $\{1, \ldots, n\}$ such that a Latin square is obtained

   in a Latin square, each integer occurs exactly once in each row and column

How to construct an LHD with n points for two continuous inputs:

1. partition experimental region into a square with n^2 cells (n along each dimension)
2. labels the cells with integers from $\{1, \ldots, n\}$ such that a Latin square is obtained

   in a Latin square, each integer occurs exactly once in each row and column
3. select one of the integers, say i, at random

How to construct an LHD with n points for two continuous inputs:

1. partition experimental region into a square with n^2 cells (n along each dimension)
2. labels the cells with integers from $\{1, \ldots, n\}$ such that a Latin square is obtained

   in a Latin square, each integer occurs exactly once in each row and column
3. select one of the integers, say i, at random
4. sample one point from each cell labelled with i

General procedure for constructing an LHD of size n given d continuous, independent inputs:

1. divide domain of each input into n intervals

General procedure for constructing an LHD of size n given d continuous, independent inputs:

1. divide domain of each input into n intervals
2. construct an $n \times d$ matrix Π whose columns are different randomly selected points permutations of $\{1, \ldots, n\}$

General procedure for constructing an LHD of size n given d continuous, independent inputs:

1. divide domain of each input into n intervals
2. construct an $n \times d$ matrix Π whose columns are different randomly selected points permutations of $\{1, \ldots, n\}$
3. each row of Π corresponds to a cell in the hyper-rectangle induced by the interval partitioning from Step 1
   sample one point from each of these cells (for deterministic inputs: centre of each cell)

Note: LHDs need not be space-filling!

Note: LHDs need not be space-filling!

Potential remedies:

- *randomised orthogonal array designs:* ensure that u-dimensional projections of points (for $u = 1, \ldots, t$) are regular grids
  exist only for certain values of n and t

Note: LHDs need not be space-filling!

Potential remedies:

- *randomised orthogonal array designs:* ensure that u-dimensional projections of points (for $u = 1, \ldots, t$) are regular grids
  exist only for certain values of n and t
- *cascading LHDs:* construct secondary LHDs for small regions around points of primary LHD

Note: LHDs need not be space-filling!

Potential remedies:

- *randomised orthogonal array designs:* ensure that u-dimensional projections of points (for $u = 1, \ldots, t$) are regular grids
  exist only for certain values of n and t
- *cascading LHDs:* construct secondary LHDs for small regions around points of primary LHD
- use additional criteria to select 'good' LHD (can also be applied to designs obtained from simple or stratified random sampling)

Distance-based designs

Key idea: Use measure of spread to assess quality of design

Distance-based designs

Key idea: Use measure of spread to assess quality of design

Examples:

- *maximin distance design:* design D that maximises smallest distance between any two points in D
  distance can be measured using L_1 or L_2 norm (or other metrics)
- *minimax distance design:* design D that minimises the largest distance between any point in the experimental region and the design

Distance-based designs

Key idea: Use measure of spread to assess quality of design

Examples:

- *maximin distance design:* design D that maximises smallest distance between any two points in D
  distance can be measured using L_1 or L_2 norm (or other metrics)
- *minimax distance design:* design D that minimises the largest distance between any point in the experimental region and the design

- *optimal average distance design:* design D that minimises average distance between pairs of points in D

  generalisation: use average distance criterion function instead of simple average of pairwise distance

- *optimal average distance design:* design D that minimises average distance between pairs of points in D

  generalisation: use average distance criterion function instead of simple average of pairwise distance

  Note: these designs need not have non-redundant projections.

  To avoid this potential problem, optimal average distance criterion can be computed for each relevant projection, and the average of these is minimised to obtain a *optimal average projection designs*.

- *optimal average distance design:* design D that minimises average distance between pairs of points in D

  generalisation: use average distance criterion function instead of simple average of pairwise distance

  Note: these designs need not have non-redundant projections.

  To avoid this potential problem, optimal average distance criterion can be computed for each relevant projection, and the average of these is minimised to obtain a *optimal average projection designs*.

  [The formulae look somewhat daunting, but are conceptually quite simple; when considering projections into subspaces with different dimensions, distances need to be normalised to make them comparable.]

Uniform Designs

Key idea: Measure uniformity of design by comparison against uniform distribution using *discrepancy measures*

Uniform Designs

Key idea: Measure uniformity of design by comparison against uniform distribution using *discrepancy measures*

Examples:

- L_∞ *discrepancy:* largest deviation between empirical distribution and uniform distribution function (= test statistic of Kolmogorov-Smirnov test for goodness of fit to uniform distribution)

  [Formal complication: cumulative empirical distribution function of vectors is based on componentwise ordering of vectors in d-dimensional space.]
- L_p *discrepancy:* average deviation distance empirical distribution and uniform distribution function, where distance is measured using an L_p norm

Uniform designs are designs with minimal discrepancy.

Uniform designs are designs with minimal discrepancy.

Uniform designs have some useful properties, *e.g.*

- for standard regression model (with known regression functions, unknown regression parameters, unknown model bias function π and normal random error, see p.144), under certain conditions on ϕ uniform designs maximise the power of the F test of regression.

Uniform designs are designs with minimal discrepancy.

Uniform designs have some useful properties, *e.g.*

- for standard regression model (with known regression functions, unknown regression parameters, unknown model bias function π and normal random error, see p.144), under certain conditions on ϕ uniform designs maximise the power of the F test of regression.
- uniform designs may often be orthogonal designs $\rightsquigarrow$ efficient algorithms for finding uniform designs may be useful in searching for orthogonal designs

Method for constructing (nearly) uniform designs:

Method for constructing (nearly) uniform designs:

Key idea: Use uniform 1-dimensional designs for each input to reduce the domain of the experimental region

Method for constructing (nearly) uniform designs:

Key idea: Use uniform 1-dimensional designs for each input to reduce the domain of the experimental region

Search over LHDs constructed from $n \times d$ matrices consisting of d permutations of $\{1, \ldots, n\}$ to find discrepancy-minimising design.

Method for constructing (nearly) uniform designs:

Key idea: Use uniform 1-dimensional designs for each input to reduce the domain of the experimental region

Search over LHDs constructed from $n \times d$ matrices consisting of d permutations of $\{1, \ldots, n\}$ to find discrepancy-minimising design.

[Fang et al. (2000) use *threshold accepting*, a stochastic local search method similar to Simulated Annealing, for solving this discrete combinatorial optimisation problem.]

Note:

- discrepancy as measured by L_∞ does not always adequately reflect our intuitive notion of uniformity (see Example 5.7, p.164*ff.*)
- other discrepancy measure may perform better [but no one seems to be sure of this]

Designs satisfying multiple criteria

- each of the the previously discussed methods and criteria produces designs with attractive properties
- **but:** none of them is completely satisfactory on their own

Designs satisfying multiple criteria

- each of the the previously discussed methods and criteria produces designs with attractive properties
- **but:** none of them is completely satisfactory on their own
- **Idea:** Generate designs that combine attractive features
- Generate and test method:
  1. generate multiple candidate designs, typically a set of LHDs
  2. select a candidate design based on a secondary criterion, *e.g.*, uniformity