Generating Optimal Latin Hypercube Designs in Real Time

By

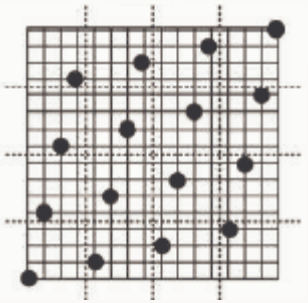
Felipe Antonio Chegury Viana
Ph.D. Student
Federal University of Uberlandia
Uberlandia, Brazil

Dr. Vladimir O. Balabanov
VisualDOC Project Manager
Vanderplaats R&D
Colorado Springs, CO, USA

Dr. Gerhard Venter
Associate Professor
Stellenbosch University
Matieland, South Africa

Dr. John Garcelon
Senior Research and Development Engigineer
Vanderplaats R&D
Monterey, CA, USA

Dr. Valder Steffen
Professor
Federal University of Uberlandia
Uberlandia, Brazil



Overview

- **Problem Area**
- **Latin Hypercube Design**
  - Optimal Latin Hypercube Design
  - Enhanced Stochastic Evolutionary Algorithm
- **Structured Latin Hypercube Design**
- **Numerical Results**
- **Conclusions**
- **Questions**

Problem Area

- **Location of Experimental Data Points**
  - Impact accuracy of generated metamodels
    - Use a reasonable number of experimental data points
- **Design of Experiments**
  - Help locate experimental data points for developing metamodels
  - Optimal Latin Hypercube (OLH) design is a popular design of experiments
    - *Generating an OLH is a time consuming optimization problem!*
- **New Methodology *(FAST)***
  - Good starting point for OLH
  - Creating good approximations to OLH

Optimal Latin Hypercube Design

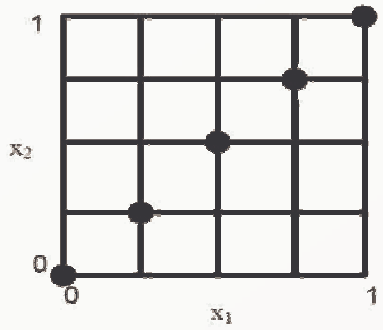
- **Popular for Computer Generated Experiments**
- **Good Space Filling Properties**
- **Model Independent**
  - Any number of design variables
- **Flexible**
  - User selectable number of experimental data points
- **Presents a Difficult Optimization Problem**
  - High computational cost for large numbers of design variables and generated points
    - Limits practical use of OLH

Latin Hypercube Design

- **Latin Hypercube Design**
  - Basis for OLH
  - Constructed for *N* design variables and for *M* experimental points by,
    - Each of the *N* variables is divided into *M* equally spaced levels
      - only one point is allowed to occupy each level
    - Often a random process is used to fill the levels
      - may result in designs with poor space filling qualities
  - OLH introduced to overcome this problem

N = 2

M = 5



OLH Theory

- **Points are pushed away from each other as much as possible**
  - Difficult, time-consuming optimization process, for example: Minimize,

$$\phi_p = \left[\sum_{i=1}^{s} J_i d_i^{-p} \right]^{1/p}$$

where,

p – positive integer (i.e., 50) for the maximum distance criteria
s – number of distinct distance values
J_i – number of pairs of points, separated by distance di
d_i – distances between points

Distances between points d_i is composed of the distinct distances between individual points, d_{ij}

$$d_{ij} = \sum_{k=1}^{N} \left| x_{ik} - x_{jk} \right|$$

where,

N – number of design variables
x_{ik} – is the k-th coordinate of the i-th point

ESEA

- **Enhanced Stochastic Evolutionary Algorithm (ESEA)**
  - Creates OLH designs
  - Relatively fast (not like GA)
  - Developed by Jinb et al. from work by Saab and Rao
- **Minimizes Φ_p**
- **Φ_p is a measure used to compare designs from our method with designs from ESEA**

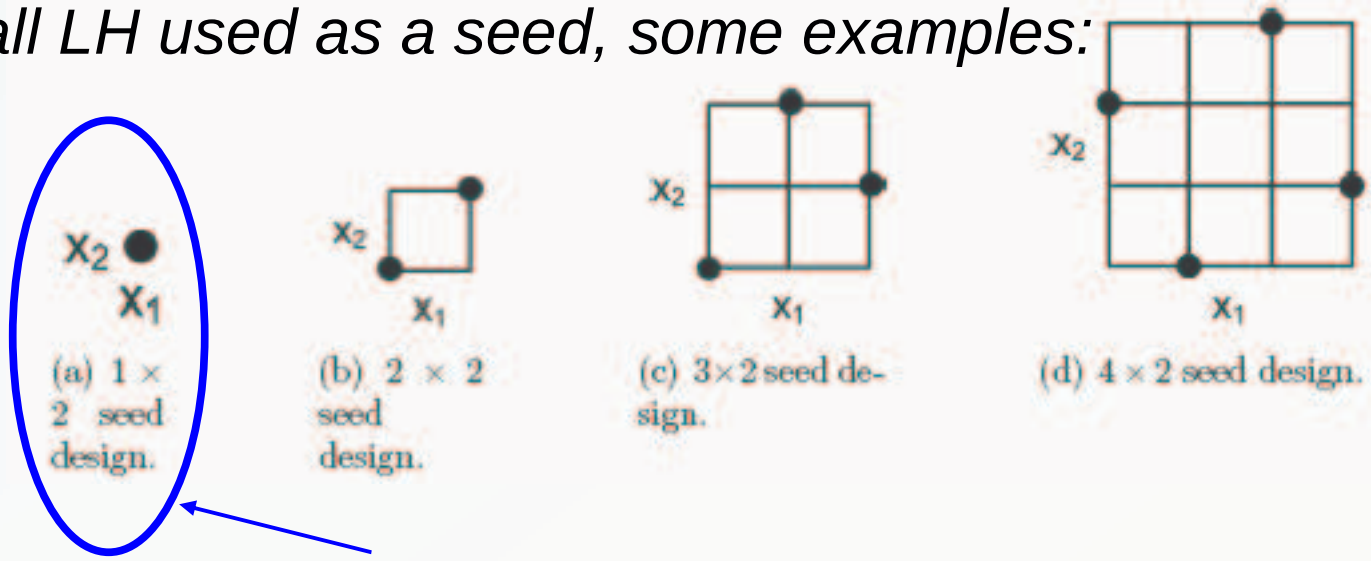
Structured Latin Hypercube

- **Empirical Approach**
  - Close to OLH without performing optimization
  - Produces a LH design with better space filling properties than LH
  - Useful standalone or as a starting point for OLH

- **Uses a Seed Design to Construct the Approximate OLH**

SLH Algorithm

- **An *N*-dimensional OLH may be constructed from an *N-dimensional seed design***
- ***Example, 16x2 OLH (16 experimental points of 2 design variables)***
  - *Small LH used as a seed, some examples:*



(a) 1 × 2 seed design.

(b) 2 × 2 seed design.

(c) 3×2 seed design.

(d) 4 × 2 seed design.

Demonstrate

SLH Algorithm (2)

- **Divide the design space into equal number of blocks**
  - Each block will be filled with the seed design

$$LH\_Size = NBlocks \times Seed\_Size$$

$$NBlocks = NDiv^{ndv}$$

$$NDiv = \sqrt[ndv]{NPnts}$$

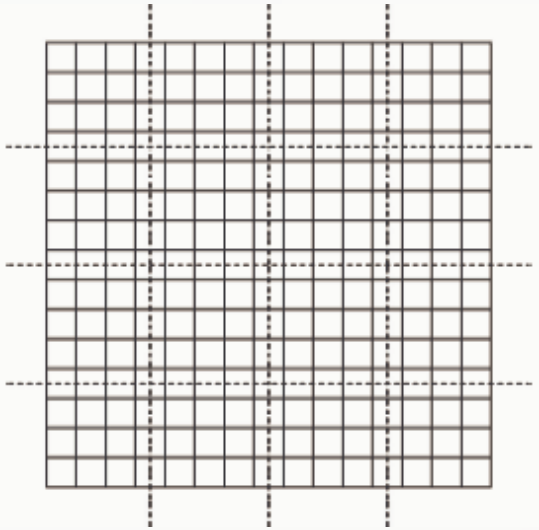
16 x 2 LH design

16 – number of experimental points (*NPnts*)

2 – number of design variables (*ndv*)

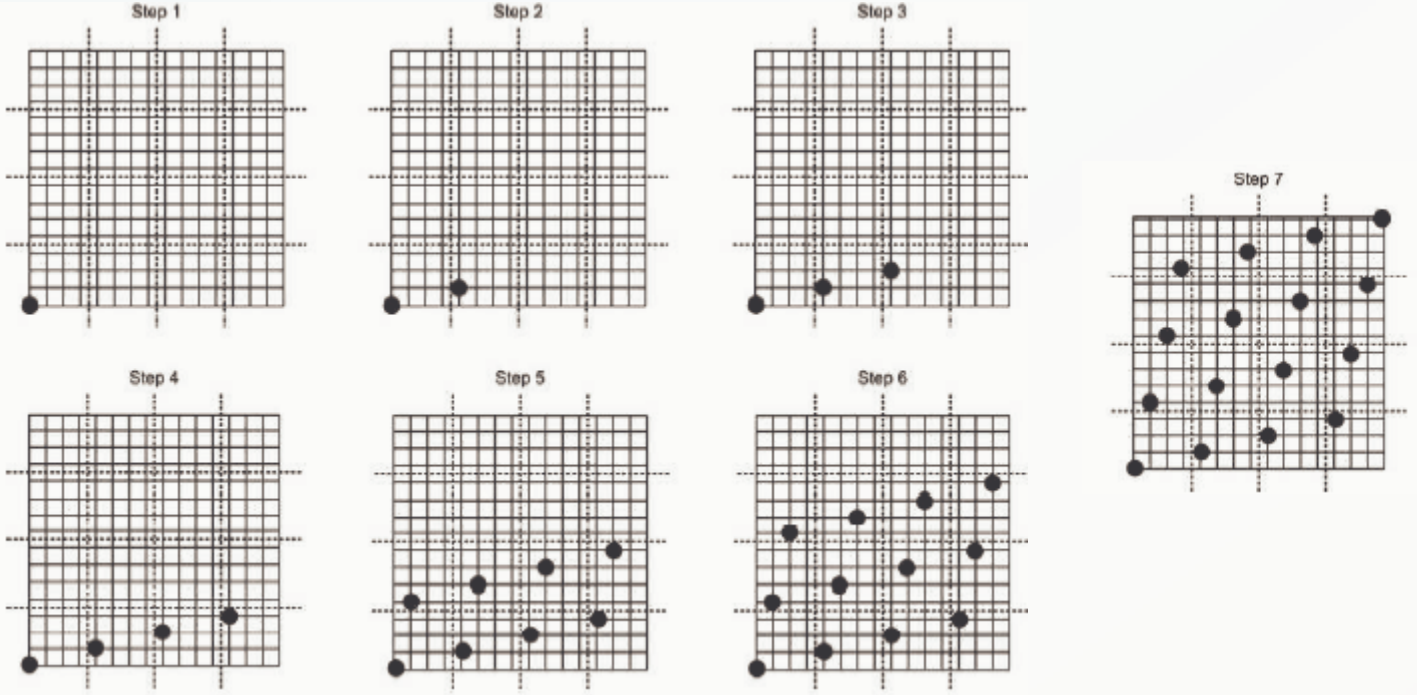
4 – number of divisions (*NDiv*)

1 – number of experimental points in the seed (*Seed_Size*)



SLH Algorithm (3)

- **Place the seed in each block**
  - Start at the origin of each block
  - Shift the origin of the seed in each block
    - Couple the row shift and column shift



Numerical Results

Optimality Criterion (Φp) Values

Design \ Approach	Worst	Random	"Optimal"	SLH
2 DV 120 pnts	0.5501	0.3333	0.0930	0.0928
2 DV 1024 pnts	0.5743	0.5070	0.0401	0.0353
3 DV 120 pnts	0.3668	0.1014	0.0334	0.0406
3 DV 512 pnts	0.3776	0.0834	0.0147	0.0171
4 DV 256 pnts	0.2793	0.0384	0.0109	0.0166
4 DV 400 pnts	0.2818	0.3755 E-1	0.6976 E-2	0.8764 E-2
5 DV 243 pnts	0.2232	0.2668 E-1	0.6842 E-2	0.1303 E-1
5 DV 500 pnts	0.2265	0.1639 E-1	0.3471 E-2	0.4334 E-2
10 DV 1024 pnts	0.1149	0.1424 E-2	0.6778 E-3	0.2195 E-2
10 DV 2000 pnts	0.1164	0.8858 E-3	0.3825 E-3	0.5945 E-4

"Optimal" values were obtained by averaging optimization results from 3 starting designs

Conclusions

- **Structured Latin Hypercube designs are good approximations to Optimal Latin Hypercube designs**
  - Requires virtually no computational time
- **For two dimensional cases, SLH cannot be improved by ESEA**
  - ESEA improves SLH for higher dimensions (at a computational cost)
- **SLH is a very good starting point for ESEA designs when the number of dimensions is high**

Questions?

Thanks for attending.