# What Are Computer Experiments and How Do We Design Them?

William Notz
Ohio State University
Columbus, OH

Joint Statistical Meetings
Seattle, WA
August 2006

## Differences between physical and computer experiments

- The code is deterministic.
  There is no random error, thus *no replication is needed.* Uncertainty is only due to a lack of knowledgde about the nature of the relationship between the inputs and the outputs.

## Differences between physical and computer experiments

- The code is deterministic.
  There is no random error, thus *no replication is needed.* Uncertainty is only due to a lack of knowledgde about the nature of the relationship between the inputs and the outputs.

- All input factors are known.
  Thus, techniques such as randomization and blocking are not needed, because there is no need to control for the effects of factors that affect the response but have not been included among the experimental factors.

## Some notation

- We assume that the code produces $r$ deterministic outputs

$$y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), \ldots, y_r(\boldsymbol{x})$$

  that depend on a set of input variables

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top$$

  For purposes of this talk, I will assume $r = 1$.

- We assume the input variables are restricted to some subset

$$\mathcal{X} \subset \mathbb{R}^d$$

and that our design consists of $n$ points

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$$

## Some desirable features of a design

- No replication.

## Some desirable features of a design

- No replication.

- Relatively few observations.

## Some desirable features of a design

- No replication.

- Relatively few observations.

- The design should cover the set of possible input values, $\mathcal{X}$, reasonably well.

  *The rationale for this is that we often don't know the nature of the functional relationship between the inputs and the outputs. Lacking such knowledge, interesting features such as maxima or minima are likely to be anywhere in* $\mathcal{X}$.

Note: Many popular "classical" designs used in response surface methodology, such as central composite designs and fractional factorial designs, do not spread points out evenly over $\mathcal{X}$. Instead they tend to place points at the boundary of $\mathcal{X}$ (corners, midpoints of faces). As a consequence, they may not be suitable for computer experiments, unless we know that a second order response surface will provide a very good approximation to the output.

## Some strategies for selecting a design

Suppose we wish to select $n$ points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ in $\mathcal{X}$ at which to observe the code. Denote the $i$-th point as

$$\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})^\top$$

For simplicity, I assume that $\mathcal{X}$ is the $d$-dimensional unit cube

$$\mathcal{X} = [0, 1]^d$$

## Sampling based designs

People typically interpret "covering the set of possible input values, $\mathcal{X}$, well" as implying that designs should spread points evenly over $\mathcal{X}$. From sampling theory, some strategies are:

## Sampling based designs

People typically interpret covering the set of possible input values, $\mathcal{X}$, well as implying that designs should spread points evenly over $\mathcal{X}$. From sampling theory, some strategies are:

- Select a random sample of $n$ points in $\mathcal{X}$.

## Sampling based designs

People typically interpret covering the set of possible input values, $\mathcal{X}$, well as implying that designs should spread points evenly over $\mathcal{X}$. From sampling theory, some strategies are:

- Select a random sample of $n$ points in $\mathcal{X}$.

- Select a stratified sample of $n$ points in $\mathcal{X}$.

## Sampling based designs

People typically interpret covering the set of possible input values, $\mathcal{X}$, well as implying that designs should spread points evenly over $\mathcal{X}$. From sampling theory, some strategies are:

- Select a random sample of $n$ points in $\mathcal{X}$.
- Select a stratified sample of $n$ points in $\mathcal{X}$.
- Select what is know as a *Latin hypercube sample* (LHS) of $n$ points in $\mathcal{X}$.

## Constructing a Latin hypercube sample (LHS)

Divide the range of each interval $[0,1]$ into $n$ subintervals of equal length. Randomly select a value from each of these subintervals. Let $x_{ij}$ (the $j$-th coordinate of $\boldsymbol{x_i}$) be the value selected for subinterval $i$ $(1 \leq i \leq n)$

*Note: It is not uncommon to take the $x_{ij}$ to be the center of the interval rather than a randomly selected point in the interval.*

Form the $n \times d$ array

$$\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ & & \vdots & \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{array}$$

Randomly permute each column, using independent permutations for each. The $n$ rows of the resulting array define the points that are our Latin hypercube sample.

Roughly speaking, a Latin hypercube sample divides each dimension of $\mathcal{X}$ into $n$ intervals. $n$ points in $\mathcal{X}$ are selected with the property that when projected onto any dimension, exactly one point is in each of the intervals for that dimension.

## Example of a five-point LHS

Assume $\mathcal{X} = [0,1]^2$. Divide each [0, 1] into the five subintervals [0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8), [0.8, 1.0].

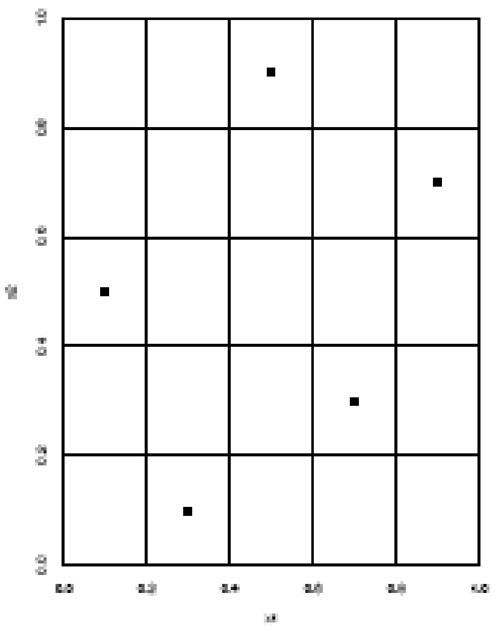
Form the $5 \times 2$ array

$$\begin{array}{cc} 0.1 & 0.1 \\ 0.3 & 0.3 \\ 0.5 & 0.5 \\ 0.7 & 0.7 \\ 0.9 & 0.9 \end{array}$$

Randomly permute each column. One possible result is

| | |
|---|---|
| 0.5 | 0.9 |
| 0.7 | 0.3 |
| 0.3 | 0.1 |
| 0.1 | 0.5 |
| 0.9 | 0.7 |

The resulting five-point LHS would be

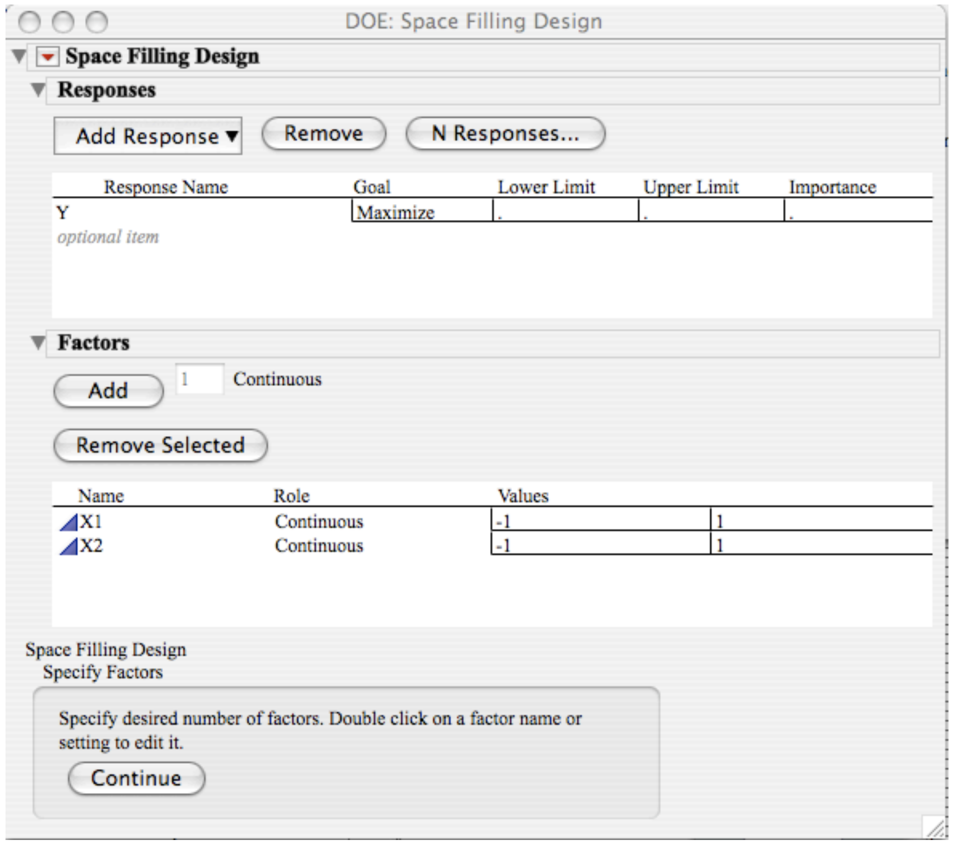


$$\Pi = \begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix}$$

You can generate LHS's using JMP.

- Under the DOE menu, select Space Filling Design.
- Add the appropriate number of factors and specify their range (Values). Click Continue.
- Select the Sample Size. Click on Latin Hypercube.

Initial space-filling design screen.



| Response Name | Goal | Lower Limit | Upper Limit | Importance |
|---|---|---|---|---|
| Y | Maximize | . | . | . |
| *optional item* | | | | |

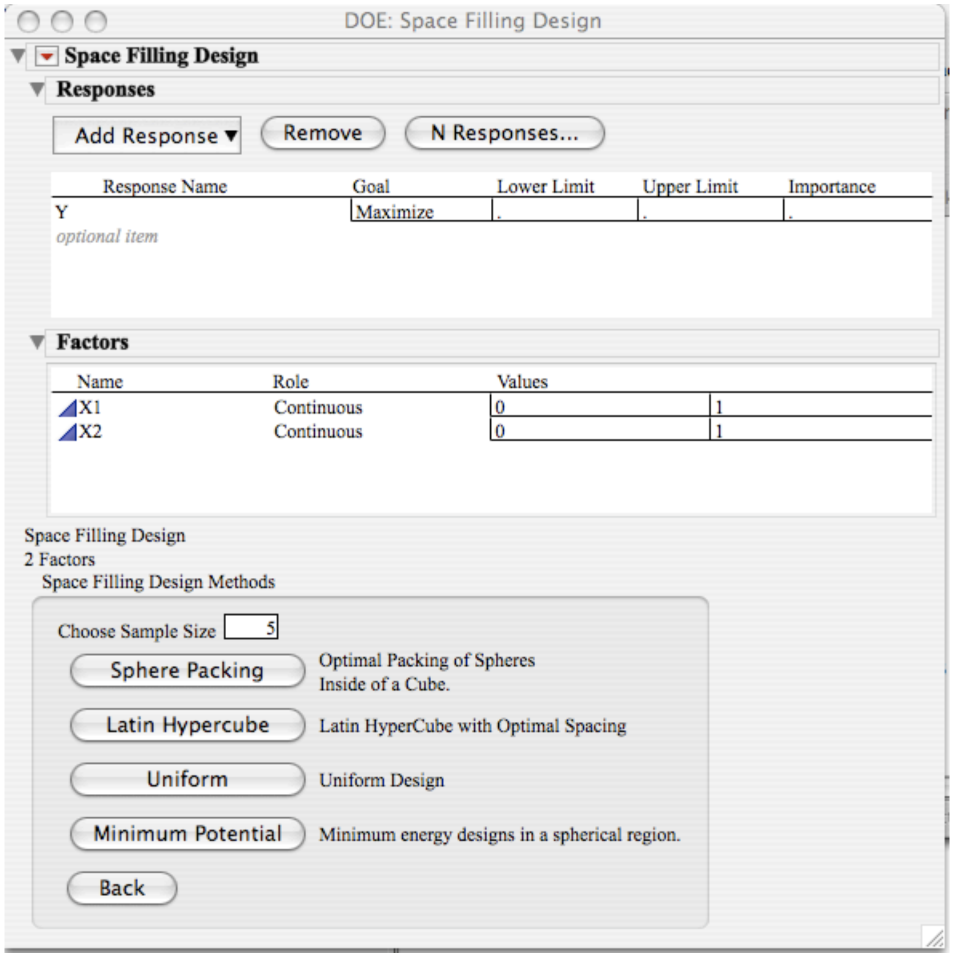
1 Continuous

| Name | Role | Values | |
|---|---|---|---|
| X1 | Continuous | -1 | 1 |
| X2 | Continuous | -1 | 1 |

Space Filling Design
Specify Factors

Specify desired number of factors. Double click on a factor name or setting to edit it.

Selecting a space-filling design.


DOE: Space Filling Design

**Space Filling Design**

**Responses**

| Response Name | Goal | Lower Limit | Upper Limit | Importance |
|---|---|---|---|---|
| Y | Maximize | . | . | . |
| *optional item* | | | | |

**Factors**

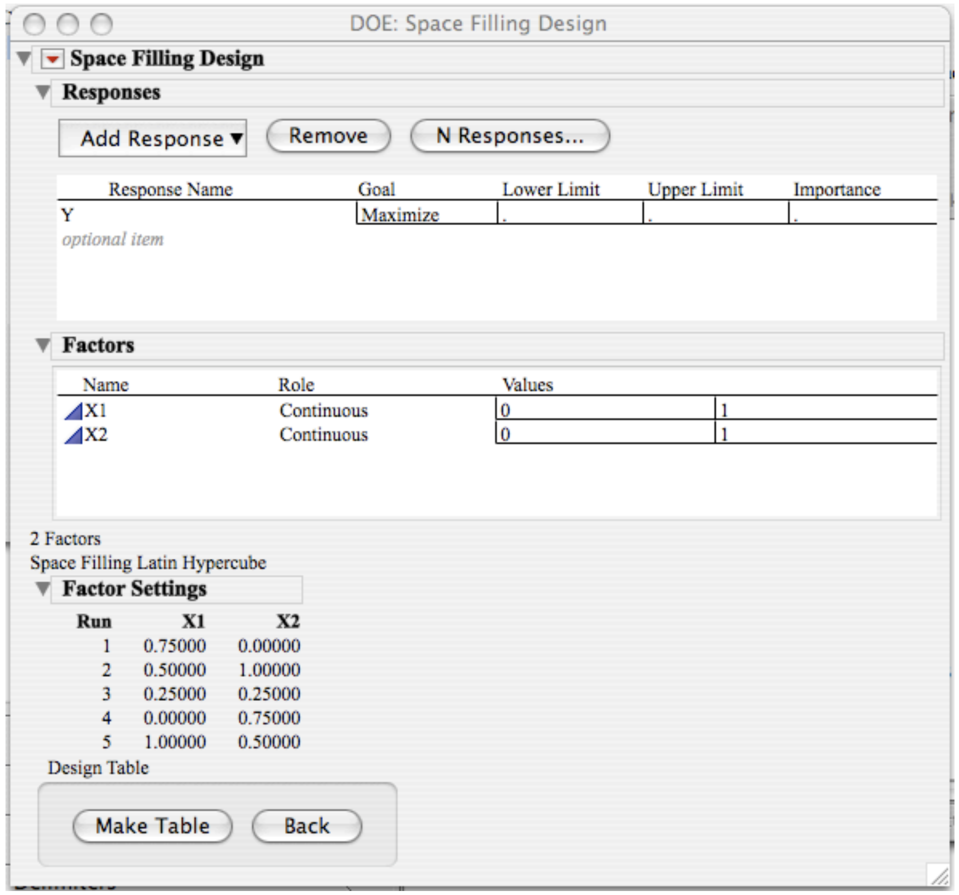
| Name | Role | Values | |
|---|---|---|---|
| X1 | Continuous | 0 | 1 |
| X2 | Continuous | 0 | 1 |

Space Filling Design
2 Factors
Space Filling Design Methods

Choose Sample Size 5

| | |
|---|---|
| Sphere Packing | Optimal Packing of Spheres Inside of a Cube. |
| Latin Hypercube | Latin HyperCube with Optimal Spacing |
| Uniform | Uniform Design |
| Minimum Potential | Minimum energy designs in a spherical region. |

The resulting design.



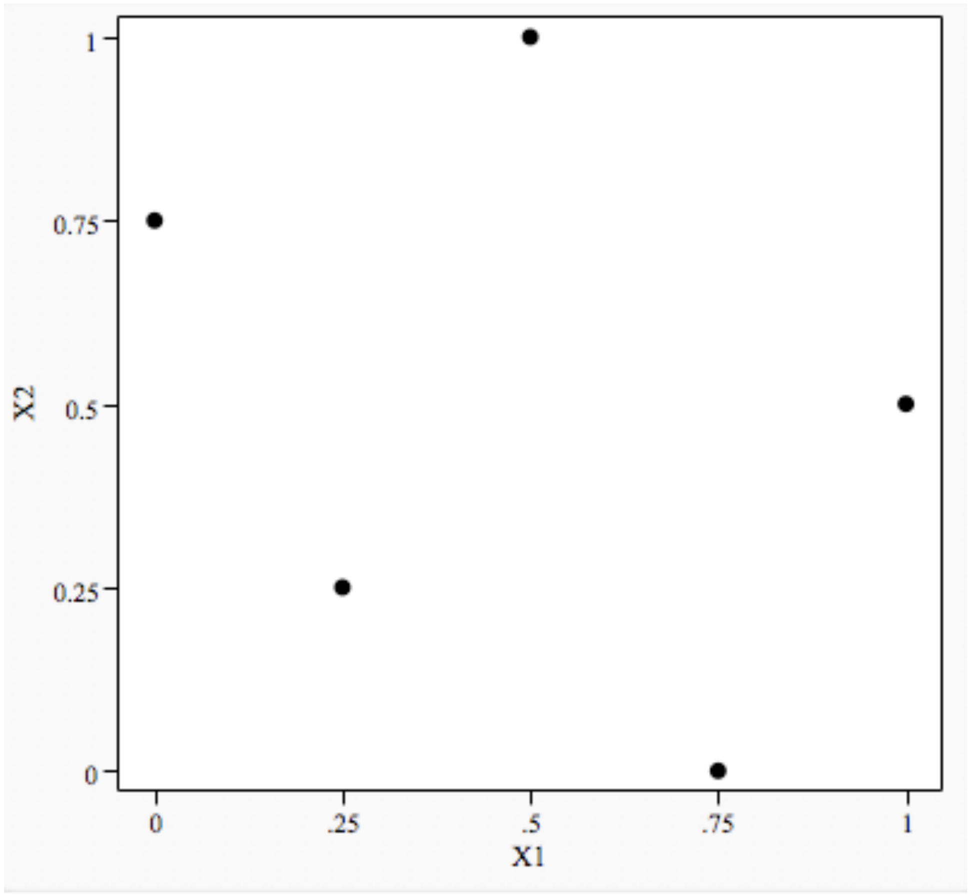
| Response Name | Goal | Lower Limit | Upper Limit | Importance |
|---|---|---|---|---|
| Y | Maximize | . | . | . |
| optional item | | | | |

| Name | Role | Values | |
|---|---|---|---|
| X1 | Continuous | 0 | 1 |
| X2 | Continuous | 0 | 1 |

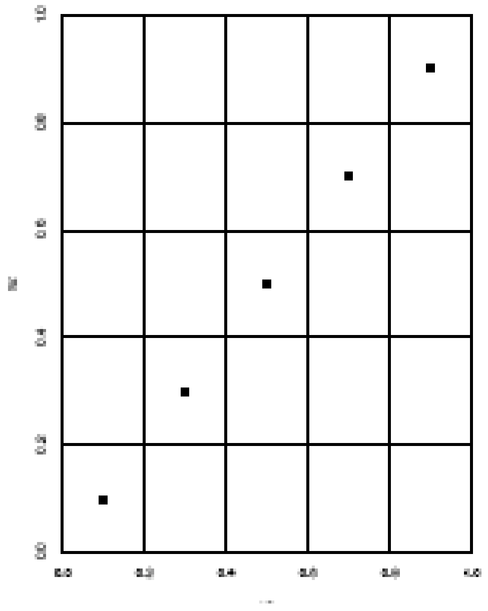
2 Factors
Space Filling Latin Hypercube

| Run | X1 | X2 |
|---|---|---|
| 1 | 0.75000 | 0.00000 |
| 2 | 0.50000 | 1.00000 |
| 3 | 0.25000 | 0.25000 |
| 4 | 0.00000 | 0.75000 |
| 5 | 1.00000 | 0.50000 |

A plot of the design.



Not all LHS's look uniformly spread out over $\mathcal{X}$. Here is a less than ideal 5 point LHS.



## Properties of LHS's

McKay, Beckman, and Conover (1979) introduced Latin hypercube sampling and subsequently other authors (for example, Stein [1987] and Owen [1992]) have explored their properties. Roughly speaking, suppose we want to find the mean of some known function $G(y(\boldsymbol{x}))$ over $\mathcal{X}$. Then the sample mean of $G(y(\boldsymbol{x}))$ computed from a Latin hypercube sample usually has smaller variance than the sample mean computed from a simple random sample.

## Problem

In the computer experiment setting we are considering, we are not usually interested in estimating the mean of some $G(y(\boldsymbol{x}))$ over $\mathcal{X}$. More typically we are interested in predicting $y(\boldsymbol{x})$ at previously unobserved inputs.

## Some comments

- When projected onto any dimension, the points in a Latin hypercube sample are spread evenly over that dimension. If $G(y(\boldsymbol{x}))$ is approximately additive, then uniformity in each dimension is intuitively appealing. This, coupled with the fact that they are relatively easy to generate, appears to account for their popularity in practice.

- There are lots of variations and generalizations of Latin hypercube sampling, including cascading Latin hypercubes, selecting from the class of all Latin hypercubes of size $n$ the one with some additional optimal property, orthogonal arrays (generalizes Latin hypercubes to designs with uniform projections onto higher dimensional subspaces).

## Distance based designs

Minimax or Maximin Designs (see Johnson, Moore, and Ylvisaker (1990) or Koehler and Owen (1996)).

A collection of $n$ points in $\mathcal{X}$ is a *minimax design* if it is a collection of $n$ points in $\mathcal{X}$ with the property that the maximum distance between this set of $n$ points and all other points in $\mathcal{X}$ is minimized.

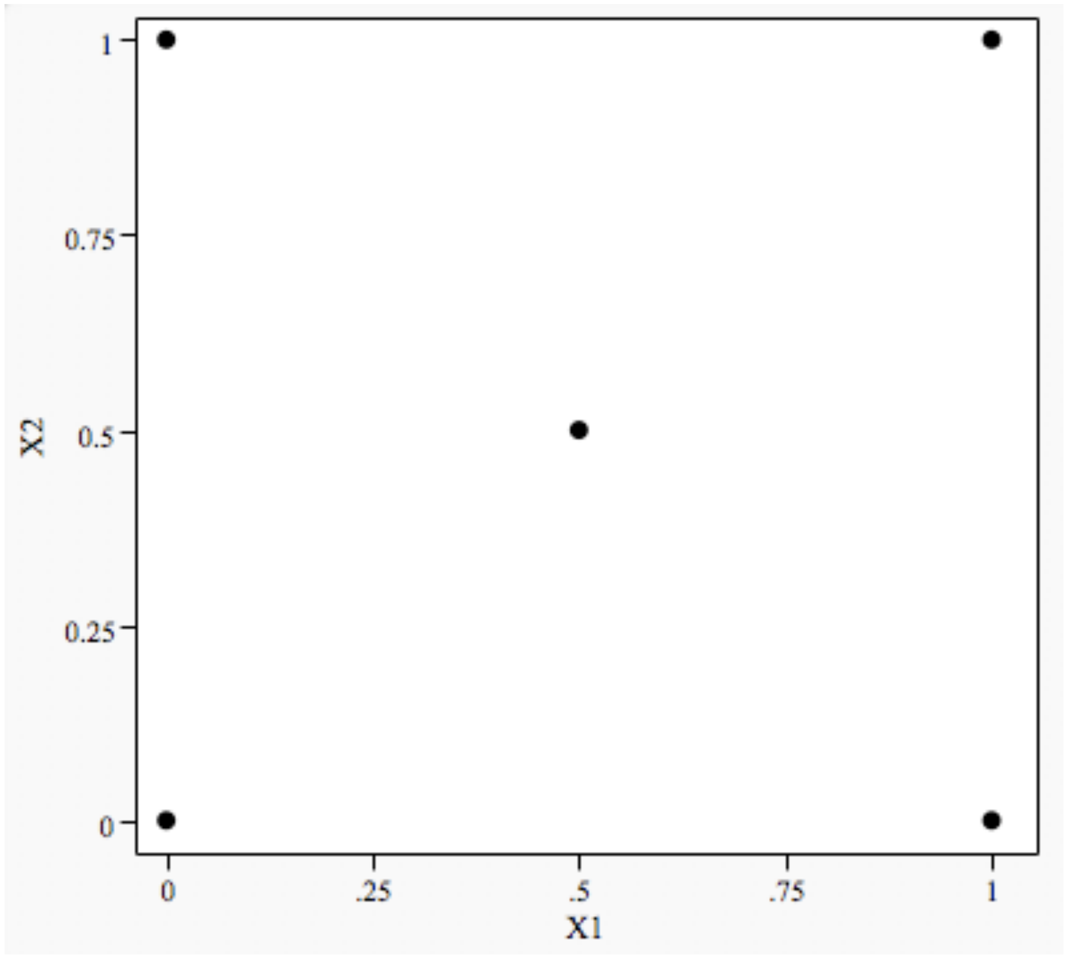
A collection of $n$ points in $\mathcal{X}$ is a *maximin design* if the minimum distance between any pair of these $n$ points is maximized.

Minimax designs are difficult to generate because distances from all other points in $\mathcal{X}$ (containing infinitely many points) must be evaluated. My impression is that very little in the way of software for generating these designs exists.

Maximin designs are relatively easy to generate. You can generate these on JMP.

- Under the DOE menu, select Space Filling Design.
- Add the appropriate number of factors and specify their range (Values). Click Continue.
- Select the Sample Size. Click on Sphere Packing.

Here is a five-point maximin design generated using JMP.



## Properties of mimimax and maximin designs

Minimax designs might be considered appealing for the following reason. Suppose we fit a model that interpolates the data (the best linear unbiased predictor of a stationary Gaussian process model has this property). Suppose that the discrepancy between our fitted model and the model generated by the computer code increases in a fixed monotonic way as the distance from a point in our design increases. Then minimax designs have a minimax property, namely they minimize the maximum (over $\mathcal{X}$) absolute difference between our fitted model and the model generated by the compute code.

Johnson, Moore, and Ylvisaker (1990) have shown that maximin designs have a sort of D-optimality property under certain conditions. These conditions assume that a certain stationary Gaussian process model is fit to the data and that the prior correlation between points is extremely weak. In fact, D-optimality is obtained in the limit as this correlation goes to 0 (so that the model looks something like the standard linear model with i.i.d. errors).

## Note

The LHS's generated by JMP are actually maximin LHS's. By this, I mean that from among all possible LHS's, JMP picks the one that is maximin. By so doing, JMP eliminates “poor” LHS's and guarantees that it generates an LHS that spreads points out reasonably evenly.

*Compared to other software packages that generate LHS's and maximin designs, JMP performs well*

## Uniform designs

Uniform designs are a collection of points that minimize some measure of discrepancy (departure or distance from a uniform distribution) and are discussed in Fang, Lin, Winker, Zhang (2000) and in the book *Design and Modeling for Computer Experiments* by Fang, Li, and Sudjianto.

To be more specific, let

$$D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$$

denote an $n$ point design. Let

$$F_n(\boldsymbol{x}) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} I[\boldsymbol{x}_i \leq \boldsymbol{x}]$$

denote the empirical cdf, where $I$ is the indicator function and the inequality is with respect to componentwise ordering of vectors in $d$-dimensional Euclidean space.

The $L_p$ *discrepancy of* $D$ is defined to be

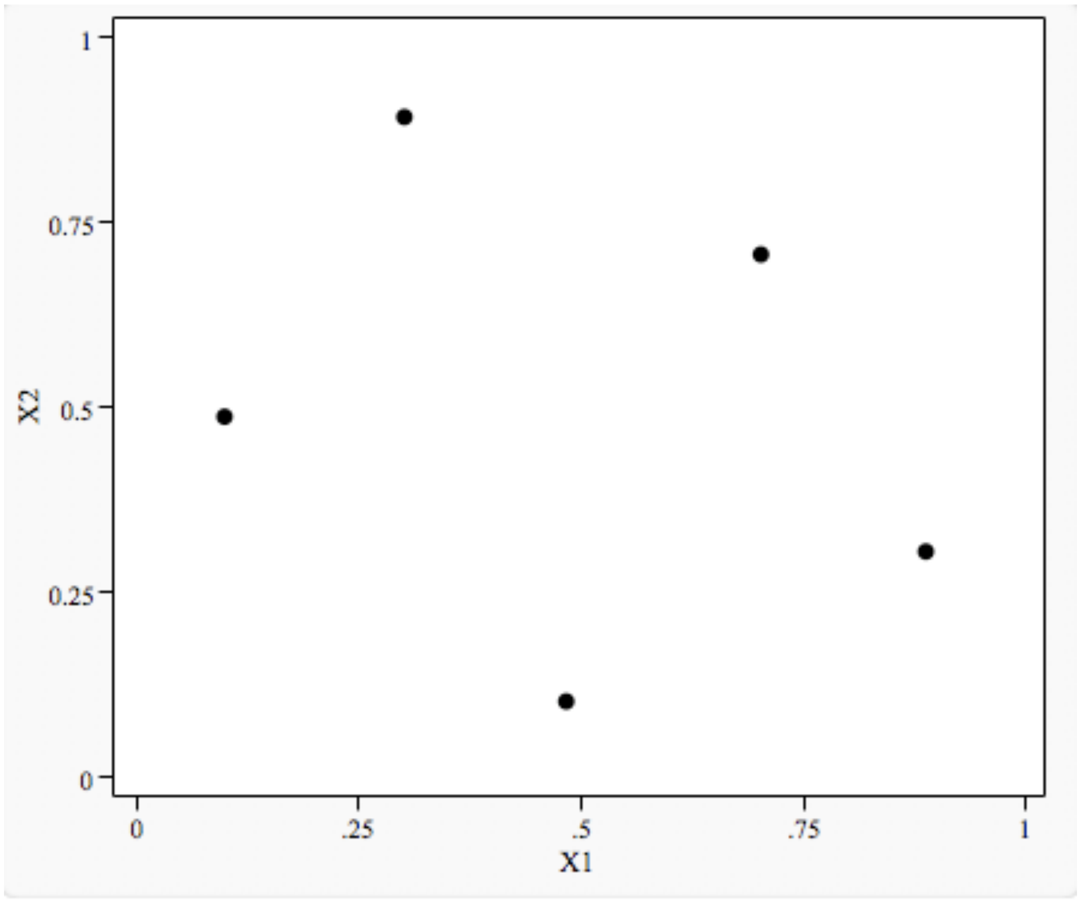
$$\left(\int_{\mathcal{X}} |F_n(\boldsymbol{x}) - F(\boldsymbol{x})|^p \, d\boldsymbol{x})^{\frac{1}{p}}\right)$$

where $F(x)$ is the uniform distribution on $\mathcal{X}$. In the limit as $p$ goes to infinity, this is called the *star discrepancy* or just the *discrepancy* of $D$.

You can generate uniform designs (with respect to the star discrepancy) in JMP.

- Under the DOE menu, select Space Filling Design.
- Add the appropriate number of factors and specify their range (Values). Click Continue.
- Select the Sample Size. Click on Uniform.

Here is a five-point uniform design generated using JMP.



## Properties of uniform designs

Suppose we want to find the mean of some known function $G(y(\boldsymbol{x}))$ over $\mathcal{X}$. Than the sample mean of $G(y(\boldsymbol{x}))$ computed from a uniform design minimizes a certain bound (the Koksma-Hlawka inequality - see Niederreiter [1992] for details) on the absolute error from the true mean.

## Problem

In the computer experiment setting we are considering we are not usually interested in estimating the mean of some $G(y(\boldsymbol{x}))$ over $\mathcal{X}$.

Note: JMP 6 has introduced another space-filling criterion, called *Minimum potential* (energy). This is appropriate for spherical regions. My (limited) understanding is that this criterion places points in a spherical region so that they balance a repulsive force that pushes the points apart with an attractive force that pulls the points towards the center of the sphere.

## Designs based on formal criteria

A number of "optimality" criteria have been proposed for designs for computer experiments. Suppose

$$\widehat{y}_D(\boldsymbol{x})$$

is our statistical predictor of $y(\boldsymbol{x})$ computed from some design $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ Let

$$\text{MSE}[\widehat{y}_D(\boldsymbol{x})]$$

denote the mean squared error of our predictor. Criteria include the following.

- The *integrated mean squared error* (IMSE) for design $D$ is proportional to

$$\int_{\mathcal{X}} \text{MSE}[\widehat{y}_D(\boldsymbol{x})]w(\boldsymbol{x})d\boldsymbol{x}$$

where $w(\boldsymbol{x})$ s some weight function on $\mathcal{X}$. A design $D^*$ that minimizes this over the class of all possible $n$ point designs on $\mathcal{X}$ is said to be *IMSE-optimal.*

Sacks, Schiller, and Welch (1989) discuss these designs.

- The *maximum mean squared error* (MMSE) for a design $D$ is proportional to

$$\max_{\boldsymbol{x} \in \mathcal{X}} \text{MSE}[\widehat{y}_D(\boldsymbol{x})]$$

A design $D^*$ that minimizes MMSE over the class of all possible $n$ point designs on $\mathcal{X}$ is said to be *MMSE-optimal.*

Sacks and Schiller (1988) discuss these designs.

- *Maximum entropy*

  If $\boldsymbol{X}$ is a random variable taking on values in $\mathcal{X}$ with pdf $f(\boldsymbol{x})$ the *entropy* of $\boldsymbol{X}$ is defined to be

$$-\int_{\mathcal{X}} f(\boldsymbol{x}) ln(f(\boldsymbol{x})) d\boldsymbol{x}$$

Entropy is a measure of the "unpredictability" of a random variable. To see this, consider the case in which $\boldsymbol{X}$ is discrete. Intuitively, if all outcomes are equally likely, then $\boldsymbol{X}$ is maximally unpredictable and a reasonable definition of entropy should assign maximal value to this distribution. If the distribution of $\boldsymbol{X}$ puts all its mass on a single value, then $\boldsymbol{X}$ is completely predictable and this should have minimum entropy.

$$-\int_{\mathcal{X}} f(\boldsymbol{x}) ln(f(\boldsymbol{x})) d\boldsymbol{x}$$

has this property.

Related to the notion of entropy is the *Shannon Information*, which is a measure of the information contained in an experiment. The Shannon Information is the negative of the entropy.

Shewry and Wynn (1987) show that if one computes the expected change in the Shannon Information from the prior distribution of a parameter to the posterior distribution, this expected change is maximized by the design that maximizes the entropy of the joint distribution of the responses given the design.

One can show that for the Gaussian process models we consider, the maximum entropy design maximizes

$$det(R)$$

where $R$ is the correlation matrix for the observations. Note that $R$ is determined by the correlation function used (including the values of any parameters that are needed to define the correlation function) and the design points.

When the correlation function is *known*, one can adapt algorithms for finding D-optimal designs to find maximum entropy designs.

## Problem

For statistical models such as stationary Gaussian process models using the best linear unbiased predictor as our statistical predictor, one needs to know the correlation parameters to compute these designs. Generally these parameters are unknown and must be estimated from data.

As a consequence, these and other criterion-based designs (for example, D-optimality) are not often used in computer experiments because they depend on unknown parameters.

Note: In practice it is possible to make reasonable prior guesses regarding the range of values of the unknown parameters for which one is likely to obtain a good-fitting model. And since fitted models appear to be somewhat insensitive to the exact values of the unknown parameters, this may make criterion-based designs not unreasonable.

Another option is to use some sort of *sequential design.* Start with an initial design, fit a model, then apply some criterion (assuming the fitted model is correct) for choosing the next point(s) in the design.

Sequential design strategies have been used for estimating the optimum of a code. See Schonlau, Welch, and Jones (1998) or Santner, Williams, and Notz (2003) for some examples.

## Designs from Monte Carlo methods

The Monte Carlo method literature includes a variety of designs that are space-filling and useful for Monte carlo integration. Designs include scrambled nets, lattice designs, and sequential designs.

A practical issue is that there does not appear to be much in the way of readily-available software to generate these designs.

Of these, one type that we have found useful are so-called *Sobol sequences.* These are space-filling designs with the property that a design with sample size $n$ is obtained from the design of sample size $n-1$ by adding a point to the design. Most space filling designs do not have this property (for example, the earlier 5-point maximin LHS generated by JMP is not obtained by adding a point to the 4-point LHS that is generated by JMP).

For details concerning the construction of Sobol sequences, download the document "Sobol sequences" by Ofelia Marin at

http://www.stat.ohio-state.edu/~comp_exp/jour.club/papers.html

R code exists for generating these sequences. To obtain and use this code, do the following.

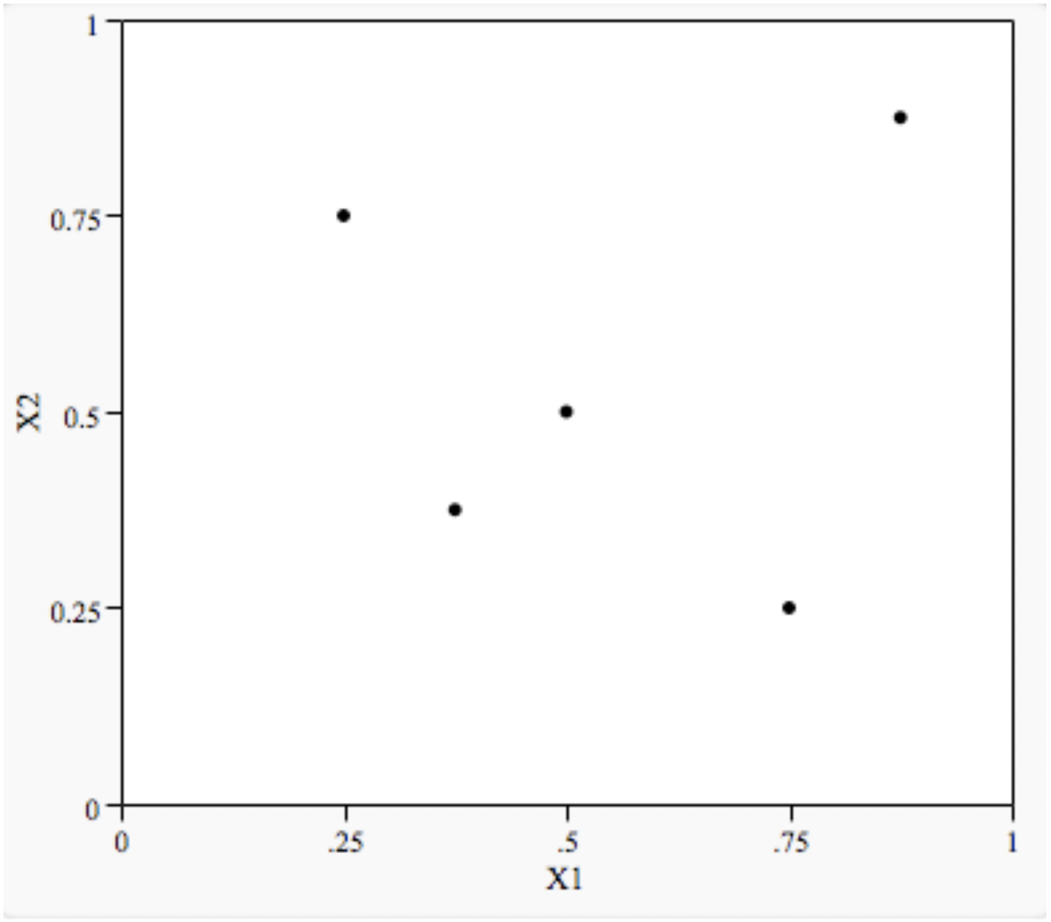
- Open R
- Use Package Installer and Package Manager (under the Packages&Data menu) to make sure that fBasics, fCalendar, fExtremes, fMultivar, fOptions, fPortfolio, and fSeries are installed and loaded.
- Use the runif.sobol command. To see how to use this, click on the fOptions package in the R Package Manager window and then on the runif.sobol link.

The basic format is

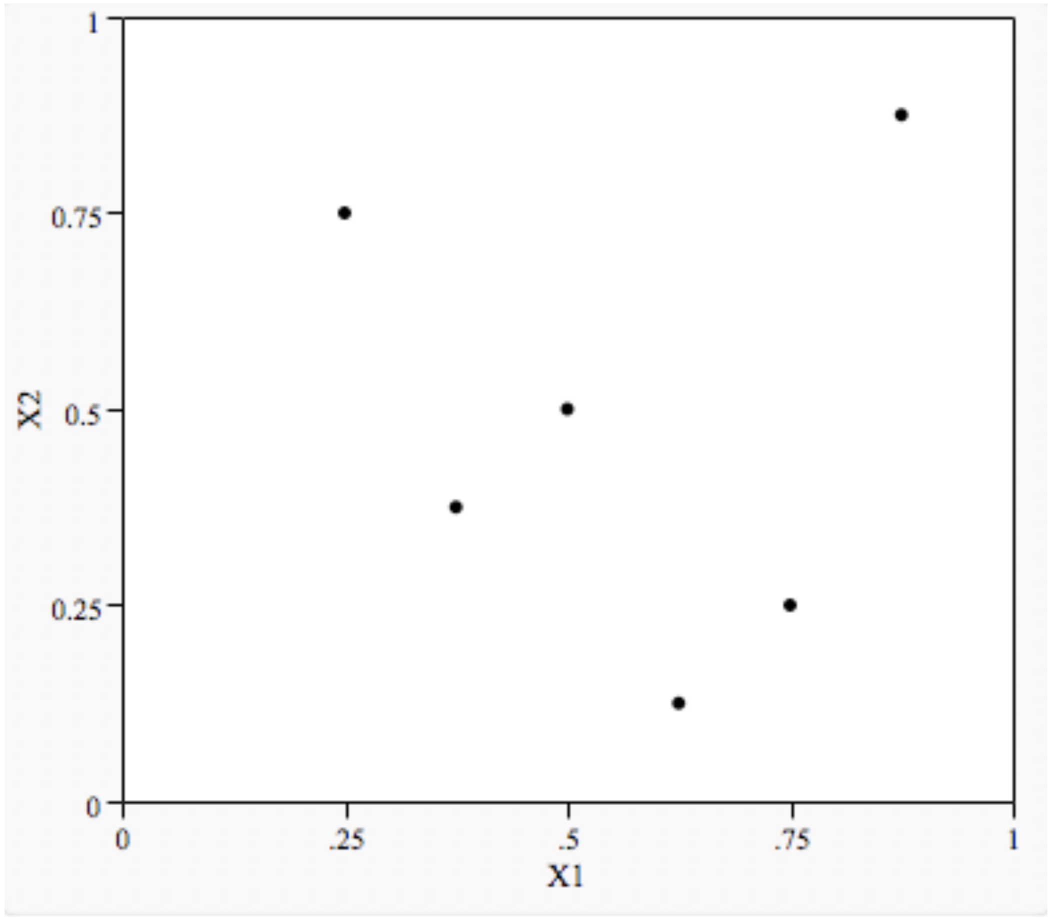
> runif.sobol([number of runs], [number of variables or dimensions])

but additional options are available.

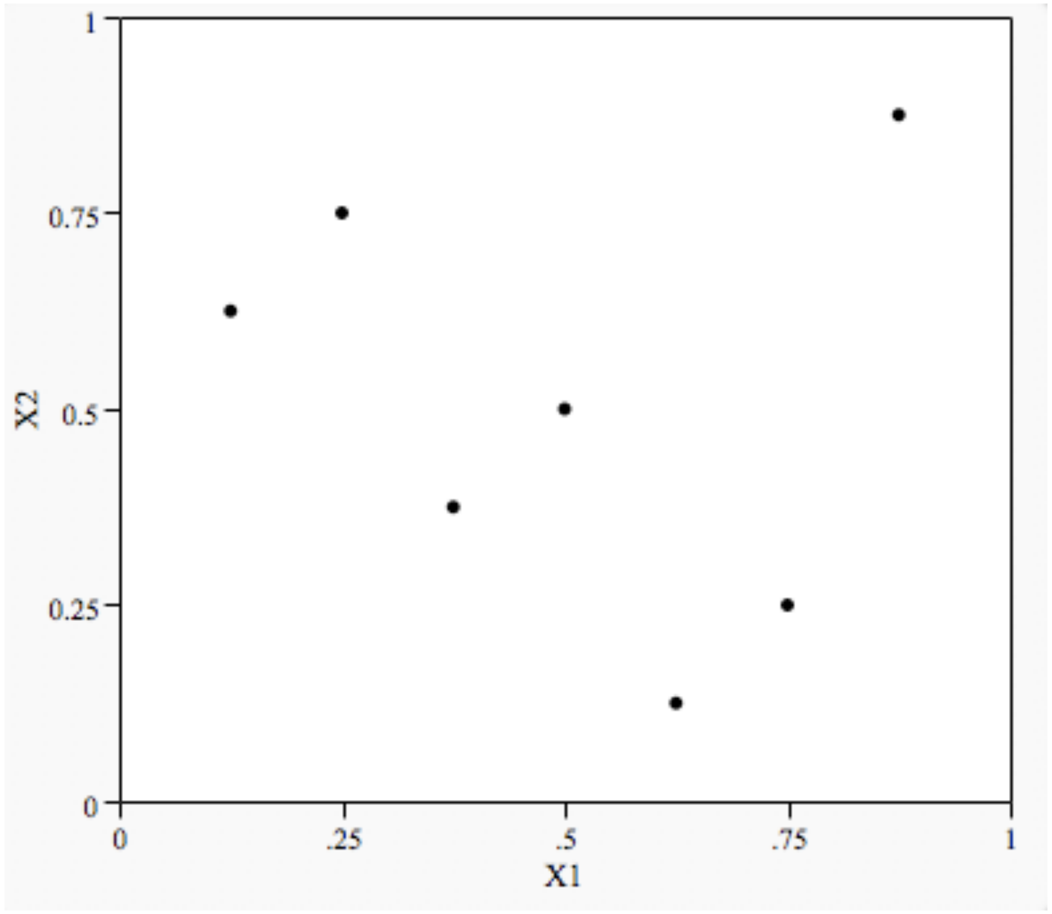
## 5-point Sobol sequence in two dimensions



6-point Sobol sequence in two dimensions



## 7-point Sobol sequence in two dimensions



## Some practical advice

- Use a space-filling design if a fixed design is required. Our experience (based on several simulations) is that there is not much difference in performance between various types of space-filling designs. However, designs that are decidedly non space-filling do not perform well.

- Sample size. There are no analytical results on sample size. A popular rule of thumb is to take about 10 observations per dimension. Thus, a $d$-dimensional problem would require $10d$ observations. We have found that fewer observations per dimension sometimes suffices (say, 5 per dimension) unless one knows in advance that the output function is quite variable over the range of the inputs.

- Sequential designs are useful if analysis can proceed sequentially and the computer code is slow. At each stage, while you wait for the code to generate the next output, you can carry out a preliminary analysis and decide if the results are sufficiently accurate.

- Experience is a great teacher. A good way to develop a feel for the models and designs we have described is to get your hands on some software for generating designs and fitting models (SAS, but also coming soon to JMP?) and then "play around" with toy examples.

Questions?

A copy of this talk is available at

http://www.stat.ohio-state.edu/∼comp_exp/jour.club/papers.html

## References

1. Fang, K.-T., Li, R., and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments* Chapman & Hall/ CRC, New York, 304pp.

2. Fang, K.-T., Lin, D. K., Winker, P., and Zhang, Y. (2000). Uniform Design: Theory and Application. *Technometrics* **42**, 237-248.

3. Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **29**, 131-148.

4. Koehler, J. R. and Owen, A. B. (1996). Computer experiments. In *Handbook of Statistics*, Vol. 13 (S. Ghosh and C. R. Rao (eds.)), pp. 261-308, Elsevier Science B. V.

5. McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**, 239-245.

6. Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia. 241pp.

7. Owen, A. B. (1992). A central limit theorem for latin hypercube samples. *Journal of the American Statistical Association* **89**, 1517-1522.

8. Sacks, J. and Schiller, S. (1988). Spatial designs. In *Statistical Decision Theory and Related Topics IV* Vol. 2 (S. S. Gupta and J. O. Berger (eds.)), pp. 385-399, Springer-Verlag, New York.

9. Sacks, J., Schiller, S. B., and Welch, W. J. (1989). Design for computer experiments. *Technometrics* **31**, 41-47.

10. Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments* Springer-Verlag, New York. 283pp.

11. Schonlau, M., Welch, W. J., and Jones, D. R. (1998). Global versus local search in constrained optimization of computer

models. In *New Developments and Applications in Experimental Design* Vol. 34 ((N. Flournoy, W. F. Rosenberger, and W. K. Wong (eds.)), pp. 11-25, Institute of Mathematical Statistics.

12. Shewry, M. C. and Wynn, H. P. (1987). Maximum entropy sampling. *Journal of Applied Statistics* **14**, 165-170.

13. Stein, M. L. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics* **29**, 143-151.