Designing Computer Experiments

Thomas Santner
The Ohio State University
Columbus, Ohio 43085

Sandia CSRI Workshop on
Mathematical Methods for Verification and Validation
August 14-16, 2007

All designs used for computer experiments can be classified into one of two classes

1. "Space-filling" designs – used for **verfication** of computer codes, as basis for producing initial estimators of **unknown** process model parameters, for screening (assuming "effect sparcity") for **quadrature** , ...
2. "Process-based" designs – used to improving **overall** prediction, for **optimization** of computer output, for **constrained optimization** of multiple computer output, for **determining** level sets, ...

Outline

I. Space-filling Designs

1. Latin Hypercube Designs (LHDs)
2. Maximum Average Distance LHDs
3. Orthogonal Array-based LHDs
4. Maximum Average Distance Symmetric LHDs
5. Quasi Monte Carlo-based Designs
6. Other criteria that lead to space-filling designs

II. Process-based Designs

1. Introduction
2. Designs for Minimizing Prediction Error
3. Designs for Optimizing Complex Computer Models

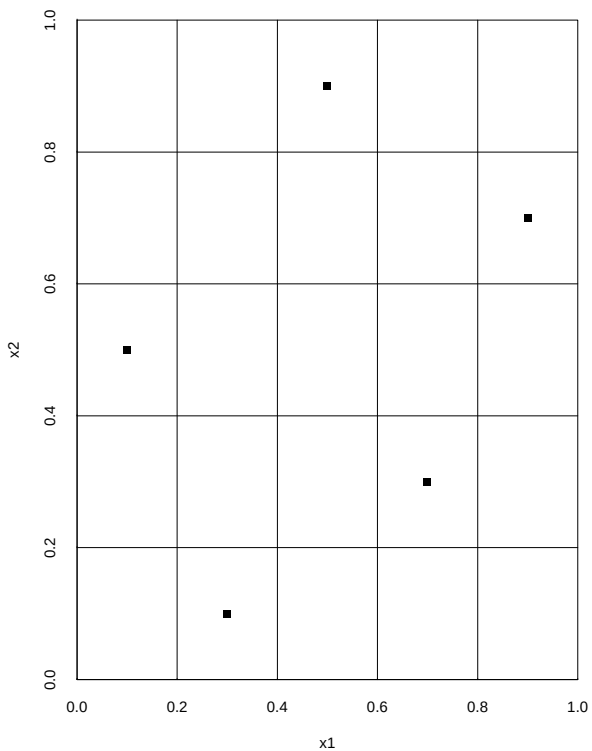
III. Summary

I. Space-filling Designs

1. Latin Hypercube Designs (LHDs)
2. Maximum Average Distance LHDs
3. Orthogonal Array-based LHDs
4. Maximum Average Distance Symmetric LHDs
5. Quasi Monte Carlo-based Designs
6. Other criteria that lead to space-filling designs

1. Latin Hypercube designs (McKay, et al., TNX, 1979)

- LHDs have uniform marginal projections
- To construct an n run LHD with p inputs, each input $\in [0, 1]$
  1. Randomly permute $1, \ldots, n$ in each of p columns
  2. rescale axis dimensions to $[0, 1]$



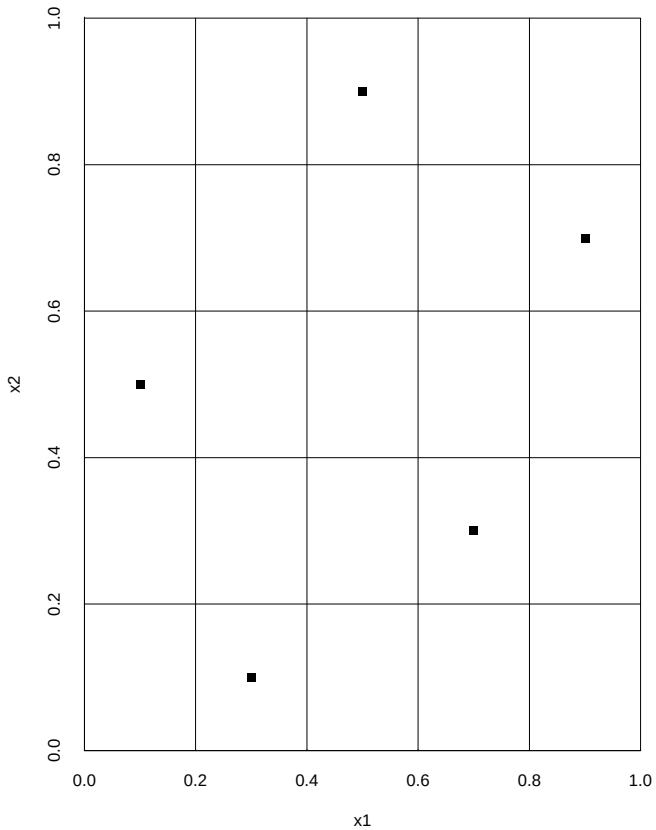
$$\begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix} \text{ or } \begin{pmatrix} .50 & .90 \\ .70 & .30 \\ .30 & .10 \\ .10 & .50 \\ .90 & .70 \end{pmatrix}$$

Critique

- Sampling-based design
- Most packages that find LHDs assume $[0, 1]^d$ inputs – rescale.

Non-rectangular regions . . .

- LHDs need not have other projections that are space-filling



2. Maximum Average Distance LHDs (Welch, TAS, 1985)

• Add distance-spreading criterion to marginal projection property of LHDs

• To construct a design with n runs in p inputs on $[0,1]^p$

1. Fix J = set of projection dimensions, e.g., $J = \{2,3\}$
2. Let $\boldsymbol{X}$ = candidate LHD, an $n \times p$ matrix
3. For $j \in J$, $\boldsymbol{X}_{j,k}$ = the $n \times j$ projection of $\boldsymbol{X}$ into k-th subset of j dimensions, $1 \le k \le \binom{p}{j}$; let $d_\rho(\boldsymbol{x}_h^{j,k}, \boldsymbol{x}_i^{j,k}) = L_\rho$ distance between $\boldsymbol{x}_h^{j,k}$ and $\boldsymbol{x}_i^{j,k}$
4. $m_{(\rho,\ell)}(\boldsymbol{X}_{j,k}) = \left(\frac{1}{\binom{n}{2}} \sum_{1 \le h < i \le n} \left[\frac{j^{1/\rho}}{d_\rho(\boldsymbol{x}_h^{j,k}, \boldsymbol{x}_i^{j,k})} \right]^\ell \right)^{1/\ell}$
   = average reciprocal distance between all row pairs of $\boldsymbol{X}_{j,k}$

Choose $\boldsymbol{X}$ to **minimize**

$$av_{(\rho,\ell)}(\boldsymbol{X}) = \left(\frac{1}{\sum_{j\in J}\binom{p}{j}} \sum_{j\in J} \sum_{k=1}^{\binom{p}{j}} \left[m_{(\rho,\ell)}(\boldsymbol{X}_{j,k}) \right]^{\ell} \right)^{1/\ell}$$

Spirit Maximize average distance between all projections of all pairs of inputs

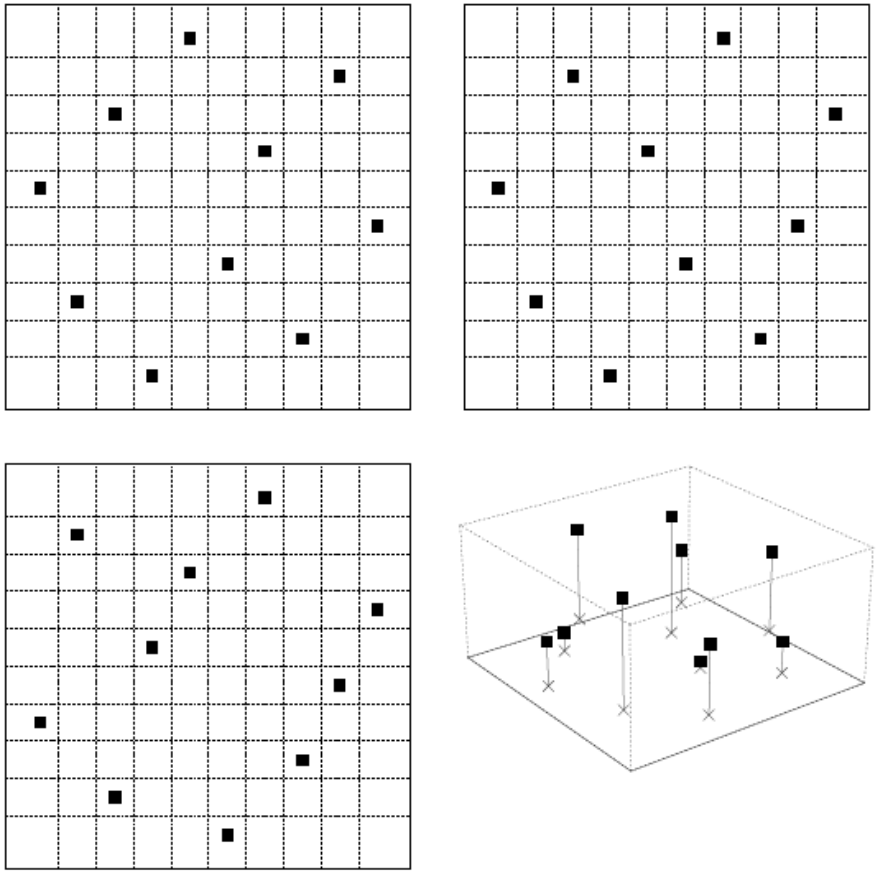
Special Case $\boldsymbol{X}^{\star} = \arg\min av_{(\rho,\ell)}(\boldsymbol{X})$ maximizes min interpoint distance as $\ell \to \infty$

Critique

- Powerful addition to the LHD criterion
- Used by ACED software

Example Maximum Average Distance LHD when

$n = 10$, $p = 3$, $\rho = 1 = \ell$, $J = \{2, 3\}$



3. Orthogonal Array-based LHDs (Tang, 1993; Owen, 1994)

• Another attempt to consider only LHDs that have built-in "good" projection properties in ≥ 2 dimensions.

• To construct an n run design in p inputs, begin with a balanced orthogonal array OA(n, p, s, t), i.e., an $n \times p$ matrix, each element of which is one of s possible values and "strength" t

1. "Usual" choice of entries is $0, \ldots, s-1$
2. **Strength** All s^t level combinations for **any** t columns occur equally often, i.e., projections of the design onto any t dimensional subspace produces points that are "spread out"
3. OA designs good for estimating output variation from **low-dimensional** effects
4. If possible, $t \geq 3$; often $s = 2$

Example Tang (1993, *Jour. Am Stat. Assoc.*) algorithm to convert OA(8,3,2,3) to an LHD

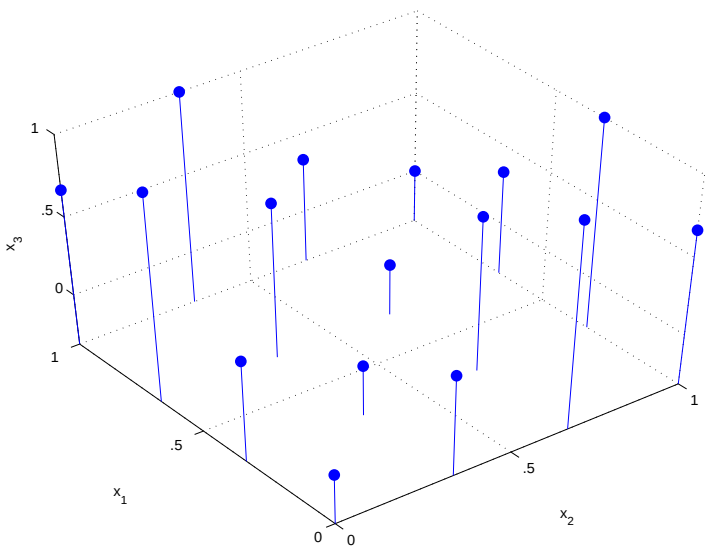
$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \rightarrow \begin{array}{ccc} 4 & 3 & 2 \\ 2 & 1 & 5 \\ 1 & 6 & 4 \\ 3 & 8 & 6 \\ 8 & 2 & 3 \\ 6 & 4 & 7 \\ 7 & 5 & 1 \\ 5 & 7 & 8 \end{array}$$

using

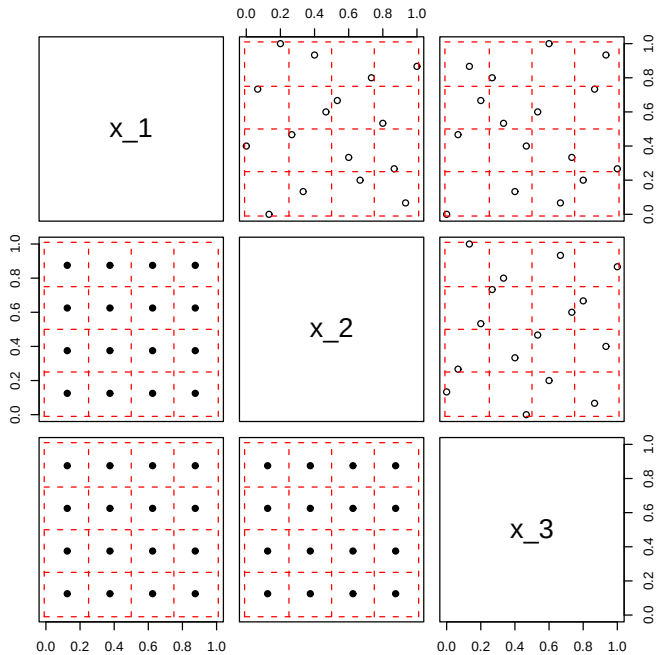
OA level	Design column	Permutation
	1	$\{4, 2, 1, 3\}$
0	2	$\{3, 1, 2, 4\}$
	3	$\{2, 4, 3, 1\}$
	1	$\{8, 6, 7, 5\}$
1	2	$\{6, 8, 5, 7\}$
	3	$\{5, 6, 7, 8\}$

Example OA-based LHD

OA(16, 3, 4, 2)



2-d projections



• Tang (1993) algorithm to convert an OA to an LHD which replaces each level of the OA in each column by a random permutation of a subset of the integers $\{1, \ldots, n\}$

$$0 \leftarrow \sigma_1(\{1, \ldots, n/s\}), \ldots, s-1 \leftarrow \sigma_s(\{(s-1)(n/s)+1, \ldots, n\})$$

• Additionally, can choose an OA-based LHD that maximizes a distance-based criterion

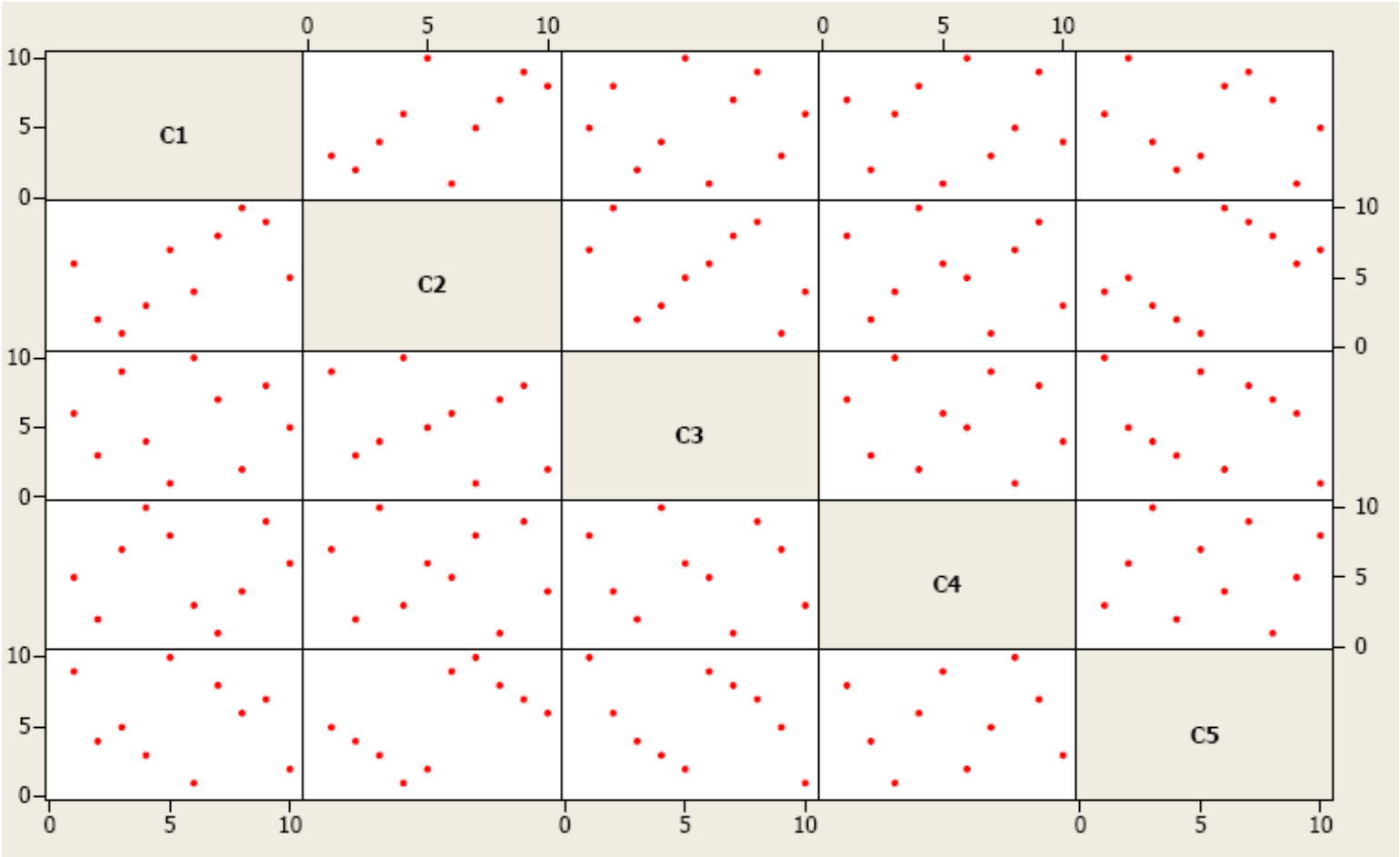
4. Max Average Distance Symmetric LHDs (Ye et al, 2000)

• Another attempt to consider only LHDs that have built-in "good" projection properties in ≥ 2 dimensions.

• Restrict attention to **symmetric** LHDs, i.e., designs such that if $(x_1, \ldots, x_p)$ is row of the $n \times p$ design then $(n+1-x_1, \ldots, n+1-x_p)$ must be another row of the design. Geometrically, plotting the rows of the design in p-space, for every row $\boldsymbol{x} = (x_1, \ldots, x_p)$ of the design, there exists another row that is the reflection of $\boldsymbol{x}$

Example of a symmetric LHD

$$\begin{pmatrix} 1 & 6 & 6 & 5 & 9 \\ 2 & 2 & 3 & 2 & 4 \\ 3 & 1 & 9 & 7 & 5 \\ 4 & 3 & 4 & 10 & 3 \\ 5 & 7 & 1 & 8 & 10 \\ 6 & 4 & 10 & 3 & 1 \\ 7 & 8 & 7 & 1 & 8 \\ 8 & 10 & 2 & 4 & 6 \\ 9 & 9 & 8 & 9 & 7 \\ 10 & 5 & 5 & 6 & 2 \end{pmatrix}$$



• **Why use SLHDs?** The estimator of the linear effects of each regression variable in a regression + stationary process model is uncorrelated with all quadratic effects and bi-linear interactions.

• To construct a design with n runs in p inputs on $[0, 1]^p$, start with a random $n \times p$ SLHD

1. Choose a symmetric LH design that maximizes some distance-based criterion (and that spreads points in two– or higher– dimensional projections)

2. Two standard search algorithms

   (a) Columnwise–pairwise algorithm [Ye, Li, and Sudjianto, 2000]

   (b) Simulated annealing algorithm [Morris, and Mitchell, 1995]

3. Repeat optimization with multiple starting designs

5. Quasi Monte Carlo-based Designs

Recall **Monte Carlo Approximation** is a numerical method used to approximate the integral

$$\int_{[0,1]^p} f(\boldsymbol{x})\, d\boldsymbol{x}. \tag{1}$$

MC chooses $\boldsymbol{X}_1, \ldots \boldsymbol{X}_N$ i.i.d uniformly distributed on $[0,1]^p$ and approximates (1) by

$$\int_{[0,1]^d} f(\boldsymbol{x})\, d\boldsymbol{x} \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{X}_i)$$

Example To approximate

$$\int_{[0,1]^5} \left(\prod_{i=1}^{5} x_i \right) \, d\boldsymbol{x} \quad \left(= \frac{1}{2^5} = 0.031250 \right)$$

```
>> X = rand(5000,5);
>> sum = 0;
>> for i=1:5000
>> sum = sum + X(i,1)*X(i,2)*X(i,3)...
              *X(i,4)*X(i,5);
>> end
>> mc_approx = sum/5000;
>> disp(mc_approx);
   0.03160440066870
```

which has a **1.134% relative error**.

Quasi-Monte Carlo Approximation is a competing method for evaluating

$$\int_{[0,1]^p} f(\boldsymbol{x})\, d\boldsymbol{x}.$$

that has better error properties than the MC approximation (the "Koksma-Hlawka Inequality"). We choose a "low-discrepancy sequence" $\boldsymbol{x}_1, \ldots \boldsymbol{x}_N$ (such as that of Soból or Niederreiter) and approximate the integral by

$$\frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i)$$

Example (continued)

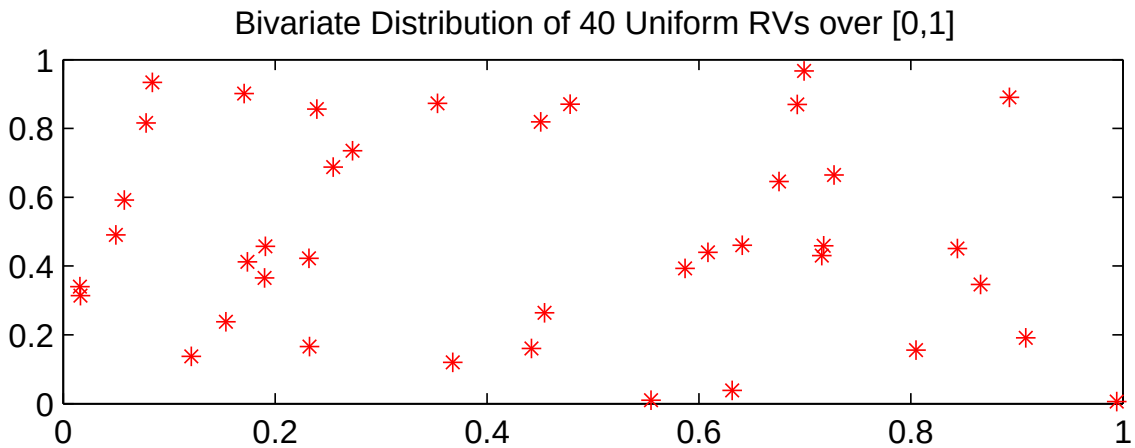
QMC approximation of $\int_{[0,1]^5} \left(\prod_{i=1}^{5} x_i\right) \, d\boldsymbol{x} = 0.031250$

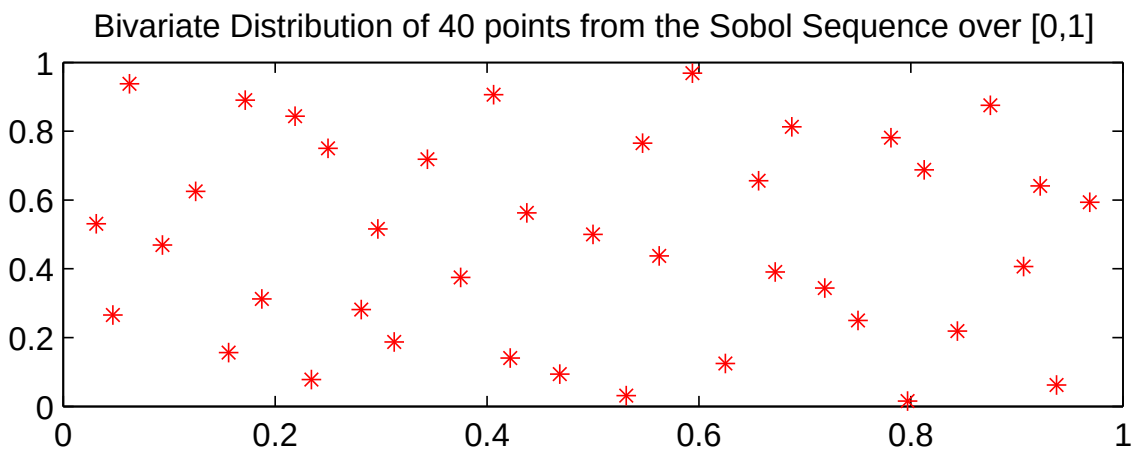
```
>> X = textread('5000by5.sobol');
>> sum = 0;
>> for i=1:5000
>> sum = sum + X(i,1)*X(i,2)*X(i,3)...
              *X(i,4)*X(i,5);
>> end
>> mc_approx = sum/5000;
>> disp(mc_approx);
   0.03123663971189
```

which has a **0.0427529219520% relative error**.

Notes

• Soból or Niederreiter QMC sequences tend to be more uniformly distributed over $[0, 1]^p$ than uniform sequences.





• In the examples studied by Morokoff and Caflisch, the QMC method did yield a more accurate result than the standard MC method with the same number of points.

• Morokoff and Caflisch found that the advantage of the QMC method is **greater** if the integrand is **smooth**, and the number of dimensions p of the integral is **small**.

• The GSL library contains programs for generating Soból and Niederreiter sequences

References Niederreiter (1992) *Random Number Generation and Quasi-Monte Carlo Methods*, Society for Industrial and Applied Mathematics.

Morokoff and Caflisch (1995) Quasi-Monte Carlo integration, *J. Comput. Phys.* **122(2)**, 218–230.

6. Other criteria that lead to space-filling designs

- **Uniform Designs**

Idea Choose n points in p inputs on $[0,1]^p$, so that their "empirical distribution function" is as close as possible to that of the $p-$dimensional uniform distribution.

Fang, Lin, Winker, Zhang (2000) "Uniform Design: Theory and Application," *Technometrics*, **42**, 237–248.

- **Low-correlation Designs**

Idea Choose n points in p inputs on $[0,1]^p$, so that columns of the design have low (zero?) correlations

A. Owen (1994) "Controlling Correlations in Latin Hypercube Samples," *Jour. Am. Statist. Assoc.*, **89**, 1517–1522.

II. Process-based Designs

1. Introduction

• These methods all assume that the true $y(\cdot)$ can be viewed as a **draw from a stochastic process** (a Bayesian viewpoint).

• The designs described below are inherently **sequential** or **multi-stage** experimental designs. Thus they are particularly useful when model calculations are **expensive**.

• If **overall prediction** of the response surface is of primary interest, sequential design based on **minimizing the mean square error** of prediction is useful.

• If **optimization** of the computer model is of interest, methods based on **expected improvement** are effective

- **A Process Model** $Y(\boldsymbol{z}) = \boldsymbol{f}^\top(\boldsymbol{z})\boldsymbol{\beta} + W(\boldsymbol{z})$
  - $\boldsymbol{f}(\boldsymbol{z}) \equiv$ regression functions, and $\boldsymbol{\beta} \equiv$ parameters
  - In computer experiments, often take $\boldsymbol{f}^\top(\boldsymbol{z})\boldsymbol{\beta} = \beta_0$
  - $W(\cdot) \equiv$ stationary, Gaussian process (GP) with mean zero, variance σ^2, and "correlation function" $R(\cdot)$ (with $R(\mathbf{0}) = 0$) so that $\mathrm{E}\{W(\boldsymbol{z})\} = 0$, $\mathrm{Var}(W(\boldsymbol{z})) = \sigma^2$, and $\mathrm{Cov}(W(\boldsymbol{z}_1), W(\boldsymbol{z}_2)) = \sigma^2 R(\boldsymbol{z}_1 - \boldsymbol{z}_2)$.
  - Thus $E\{Y(\boldsymbol{z})\} = \boldsymbol{f}^\top(\boldsymbol{z})\boldsymbol{\beta}$, $\mathrm{Var}(Y(\boldsymbol{z})) = \sigma^2$, and $\mathrm{Cov}(Y(\boldsymbol{z}_1), Y(\boldsymbol{z}_2)) = \sigma^2 R(\boldsymbol{z}_1 - \boldsymbol{z}_2)$.
- **Notation** Given inputs $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ let $\boldsymbol{y}^n = (y(\boldsymbol{z}_1), \ldots, y(\boldsymbol{z}_n))^\top$ denote the corresponding computer outputs

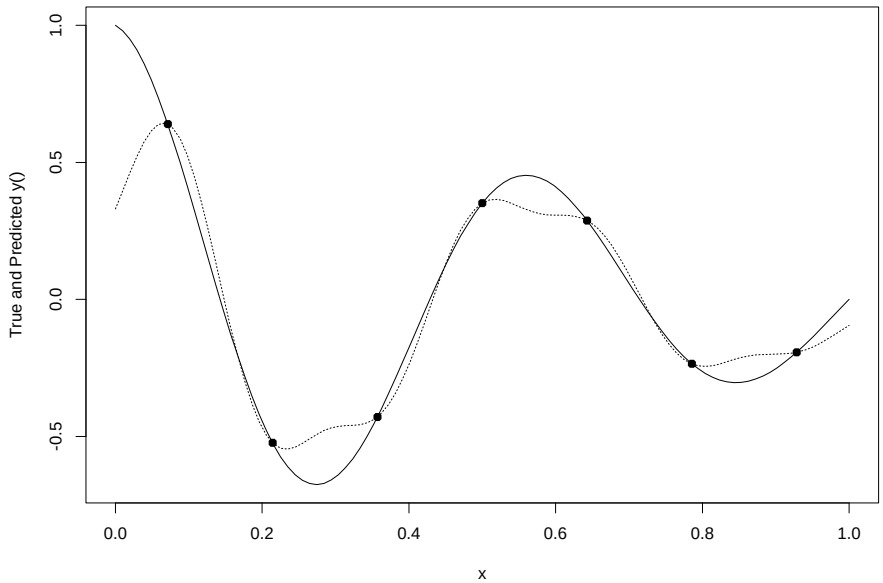
- **Application #1 of the Process Model** Assuming a noninformative prior for $\boldsymbol{\beta}$ and completely specified covariance for $Y(\boldsymbol{z})$ the **predictive distribution** of $Y(\boldsymbol{z})$ is given by

$$[Y(\boldsymbol{z})|\boldsymbol{Y}^n = \boldsymbol{y}^n] \sim N\left(\widehat{y}(\boldsymbol{z}), s^2(\boldsymbol{z})\right), \text{ where}$$

$$\begin{aligned}\widehat{y}(\boldsymbol{z}) &= E\left\{Y(\boldsymbol{z})|\boldsymbol{Y}^n = \boldsymbol{y}^n\right\} \\ &= \boldsymbol{f}^T(\boldsymbol{z})\widehat{\boldsymbol{\beta}} + \boldsymbol{r}^T(\boldsymbol{z})\boldsymbol{R}^{-1}\left(\boldsymbol{y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right)\end{aligned}$$

- $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{z}) = (R(\boldsymbol{z} - \boldsymbol{z}_1), \ldots, R(\boldsymbol{z} - \boldsymbol{z}_n))^\top$
- $\boldsymbol{R} = (\, R(\boldsymbol{z}_i - \boldsymbol{z}_j)\,)$, $i, j = 1, \ldots, n$
- $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^\top \boldsymbol{R}^{-1}\boldsymbol{F}\right)^{-1} \boldsymbol{F}^\top \boldsymbol{R}^{-1}\boldsymbol{y}^n$ (GLS estimate of $\boldsymbol{\beta}$)
- $\boldsymbol{F} = [\boldsymbol{f}(\boldsymbol{z}_1); \cdots ; \boldsymbol{f}(\boldsymbol{z}_n)]^\top$

Example



Notes

- $\widehat{y}(\boldsymbol{z})$ Interpolates data
- Stationarity (constant variance,. . .) ? Stationarity dictates extrapolation curve

while

$$s^2(\boldsymbol{z}) = E\left\{(Y(\boldsymbol{z}) - \widehat{y}(\boldsymbol{z}))^2 | \boldsymbol{Y}^n = \boldsymbol{y}^n\right\}$$
$$= \sigma^2 \left\{1 - \boldsymbol{r}^\top \boldsymbol{R}^{-1} \boldsymbol{r} + \boldsymbol{h}^\top \left(\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{h}\right\}$$

is a measure (almost) of the uncertainty in the predictor where $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{z}) = \boldsymbol{f}(\boldsymbol{z}) - \boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{z})$

2. Designs for Minimizing Prediction Error

- As usual, assume the $p \times 1$ input $\boldsymbol{z}$ has been scaled to $[0,1]^p$. Let $\mathcal{D} = \mathcal{D}_n = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ denote an n-point design.

- Define the **integrated mean square prediction error (IMSPE) criterion** as follows:

$$J(\mathcal{D}) = \int_{[0,1]^p} \frac{1}{\sigma^2} E\left\{(Y(\boldsymbol{z}) - \widehat{y}(\boldsymbol{z}))^2 | \boldsymbol{Y}^n = \boldsymbol{y}^n\right\} \, w(\boldsymbol{z}) \, d\boldsymbol{z}$$

  - $E\left\{(Y(\boldsymbol{z}) - \widehat{y}(\boldsymbol{z}))^2 | \boldsymbol{Y}^n = \boldsymbol{y}^n\right\} \equiv$ MSPE of $\widehat{y}(\boldsymbol{z})$
  - Thus $J(\mathcal{D})$ is the MSPE of $\widehat{y}(\boldsymbol{z})$ "averaged" over $[0,1]^p$ w.r.t. weight $w(\boldsymbol{z})$

- The n-point design $\mathcal{D}_{imspe}$ is **IMSPE-optimal** if it **minimizes** the IMSPE criterion function, i.e.,

$$J(\mathcal{D}_{imspe}) = \min_{\mathcal{D}} J(\mathcal{D})$$

  - Under simplifying assumptions that often occur in practice, an expression for $J(\mathcal{D})$ can be derived that allows for straightforward computation.
  - The IMSPE criterion is related to **L-optimality**

- The **maximum mean square prediction error (MMSPE) criterion** is defined to be

$$Q(\mathcal{D}) = \max_{\boldsymbol{z} \in [0,1]^p} \frac{1}{\sigma^2} \text{MSPE}(\widehat{y}(\boldsymbol{z}))$$

- The n-point design $\mathcal{D}_{mmspe}$ is **MMSPE-optimal** if

$$Q(\mathcal{D}_{mmspe}) = \min_{\mathcal{D}} Q(\mathcal{D})$$

  - Related to **G-optimality**

- **In Practice** The IMSPE and MMSPE criterion functions depend on unknown GP correlation parameters, say $\boldsymbol{\xi}$. Therefore it is difficult to implement these criteria in a single-stage procedure. In this case, a sequential procedure is natural. For example,

  S0 Start with an initial n-run space-filling design $\mathcal{D}_0$, such as a Maximin LHD. Estimate $\boldsymbol{\xi}$ using ML or REML, say, giving $\widehat{\boldsymbol{\xi}}_n$.

  S1 Augment $\mathcal{D}_0$ with g candidate points, $g \geq 1$, $\mathcal{D} = (\mathcal{D}_0, \boldsymbol{z}_{n+1}, \ldots, \boldsymbol{z}_{n+g})$. Optimize $J(\mathcal{D})$ or $Q(\mathcal{D})$ using $\widehat{\boldsymbol{\xi}}_n$ in place of $\boldsymbol{\xi}$.

  S2 Repeat *S1* as desired, updating the $\boldsymbol{\xi}$ correlation parameter estimates after each stage.

3. Designs for Optimizing Complex Computer Models

(Schonlau, 1997; Jones et al., 1998)

- **Goal** Find $\boldsymbol{z}^\star = \text{argmin}_{\boldsymbol{z}\in[0,1]^p}\ y(\boldsymbol{z})$ (especially when computer model evaluations are **expensive**)
- Assume computer model output is a realization of the GaSP

$$Y(\boldsymbol{z}) = \boldsymbol{f}^\top(\boldsymbol{z})\boldsymbol{\beta} + W(\boldsymbol{z})$$

- **Nomenclature** Expected Improvement Algorithms or Efficient Global Optimization (EGO) Algorithms

An Expected Improvement Algorithm

- **Overview**

  1. Compute $y(\cdot)$ at the points in an initial (space-filling) design.
  2. Use the information from the initial runs to select the next point(s) $\boldsymbol{z}$ to evaluate the code according to an expected improvement criterion.
  3. Continue selecting points using the information from all of the previous runs until a stopping criterion is met.

- **Training Design/Data** $\mathcal{D}_n = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ and associated $\boldsymbol{y}^n = (y(\boldsymbol{z}_1), \ldots, y(\boldsymbol{z}_n))$

- **(Theoretical) Improvement** Let $y^{\min} = \min\{y(\boldsymbol{z}_i) : \boldsymbol{z}_i \in \mathcal{D}_n\} =$ best $y(\cdot)$ computed after n output evaluations. For arbitrary $\boldsymbol{z}$,

$$i_n(\boldsymbol{z}) = \begin{cases} y^{\min} - y(\boldsymbol{z}), & y^{\min} > y(\boldsymbol{z}) \\ 0, & \text{o.w.} \end{cases},$$

Not computable because $y(\boldsymbol{z})$ is not known

- **Probabilistically-based Improvement** For arbitrary $\boldsymbol{z}$,

$$I_m(\boldsymbol{z}) = I_m(\boldsymbol{z}; \sigma^2, \boldsymbol{\xi}) \quad = \quad \begin{cases} y^{\min} - Y(\boldsymbol{z}), & y^{\min} > Y(\boldsymbol{z}) \\ 0, & \text{ow} \end{cases}$$

where the *unknown* code value $y(\boldsymbol{z})$ is replaced by its (random function) prior $Y(\boldsymbol{z})$, and $(\sigma^2, \boldsymbol{\xi})$ denotes the variance and correlation parameters of the $W(\cdot)$ process.

An Expected Improvement Algorithm (in practice)

S0: Choose the **initial** set of design points $\mathcal{D}_n = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$

S1: **Estimate** the variance and correlation parameters of the $W(\cdot)$ process by ML or REML, say $\widehat{\sigma_n^2}$ and $\widehat{\boldsymbol{\xi}}_n$.

S2: Choose the $(n+1)$-st input $\boldsymbol{z}_{n+1}$ to maximize the *posterior expected improvement* given the data, $\widehat{\sigma_n^2}$ and $\widehat{\boldsymbol{\xi}}_n$, i.e.,

$$E\left\{ I_n(\boldsymbol{z}) | \boldsymbol{y}^n, \widehat{\sigma_n^2}, \widehat{\boldsymbol{\xi}}_n \right\} .$$

S3: If the stopping criterion **is not met**, set $S_{n+1} = S_n \cup \boldsymbol{z}_{n+1}$, calculate $y(\boldsymbol{z}_{n+1})$, increment n to $(n+1)$, and go to *S1*. If the criterion **is met** , predict the global minimizer to be an element of S_n with code value y^{min}.

Computation of Posterior Expected Improvement

- The posterior expected improvement has the explicit closed-form

$$E\left\{I_m(\boldsymbol{z})|\boldsymbol{y}^n\right\} =$$

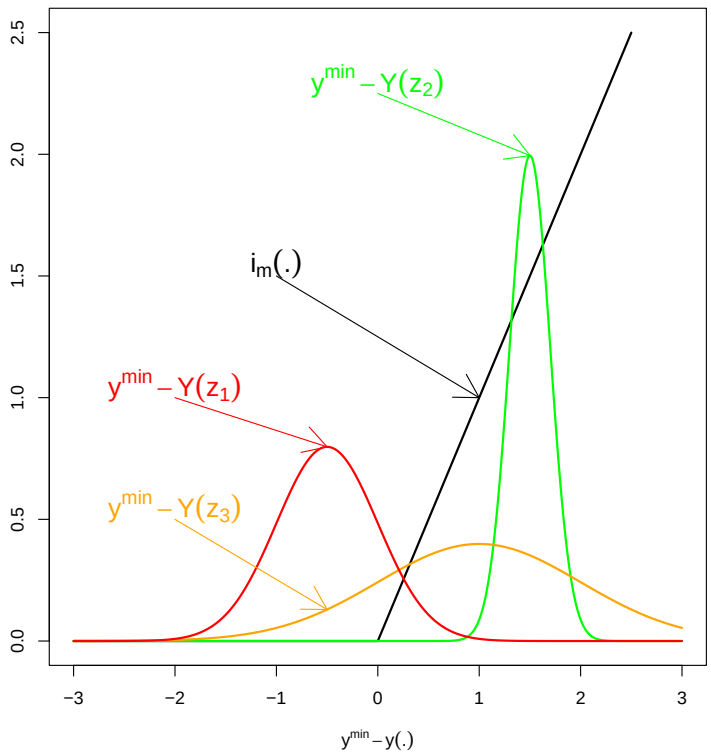
$$\left(y^{min} - \widehat{y}(\boldsymbol{z})\right)\Phi\left(\frac{y^{min} - \widehat{y}(\boldsymbol{z})}{s(\boldsymbol{z})}\right) + s(\boldsymbol{z})\phi\left(\frac{y^{min} - \widehat{y}(\boldsymbol{z})}{s(\boldsymbol{z})}\right),$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the $N(0,1)$ cdf and pdf.

- LH term of posterior EI is "large" for $\boldsymbol{z}$ having a predicted value that is much smaller than the best minimum computed so far, i.e., $\widehat{y}(\boldsymbol{z}) \ll y^{min}$ ("local" search)

- RH term of posterior EI is large when there much uncertainty about the value of $y(\boldsymbol{z})$, i.e., when $s(\boldsymbol{z})$ is large ("global" search)

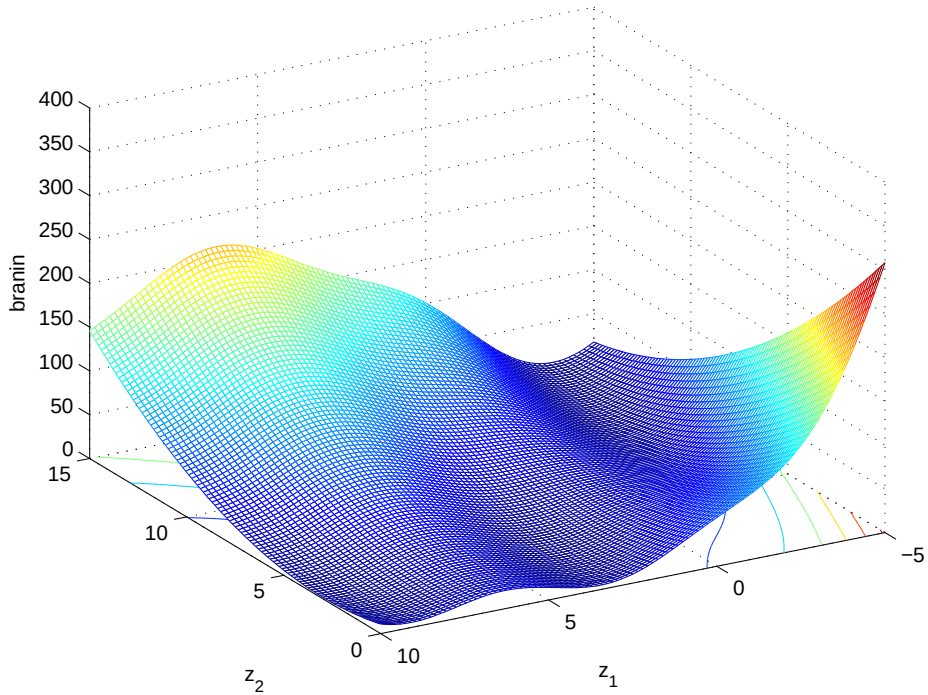
- **Graphical look at improvement**



1. z_1: small improvement
2. z_2: guaranteed improvement (local search)
3. z_3: possibility of large improvement (global search)

An Example

- **Two inputs** with $\boldsymbol{z} = (z_1, z_2) \in [-5, 10] \times [0, 15]$
- **Objective Function** : Branin function



- Three global minima:

$$y(\pi, 2.275) = y(3\pi, 2.475) = y(-\pi, 12.275) = 0.39789.$$

EI Algorithm Implemented in SPACE Software

- SPACE available from `http://www.schonlau.net/`

```
X < xstart.mat
Y < ystart.mat

Tries = 5
Fit

ObjectiveFunction = branin

XDescription < xdescrip.mat
Minimize

X > xfinal.mat
Y > yfinal.mat
```

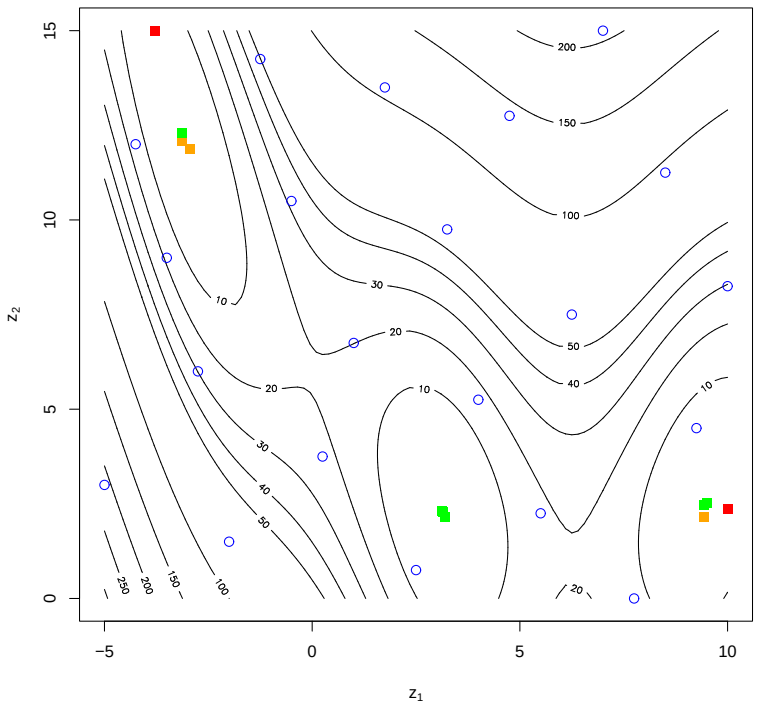
SPACE jobfile

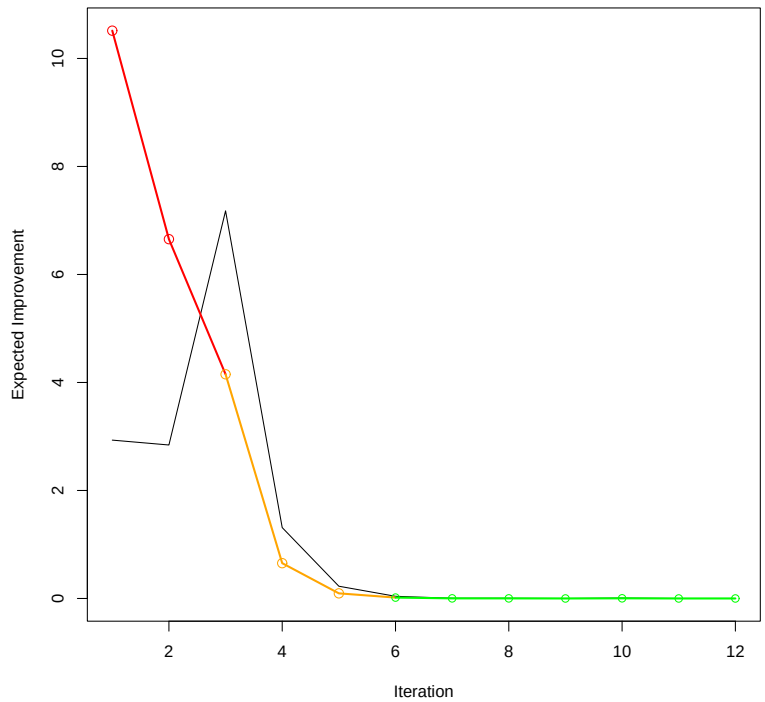
```
#! /usr/bin/perl

$IN = $ARGV[-1];
open IN, $IN;

$xx = <IN>;
chomp $xx; @x=split " ", $xx;
$n = $x[0]; shift @x;
close IN;

$PI = 3.141593;
$y = ($x[1]-(5.1/(4*$PI**2))*$x[0]**2+
     (5/$PI)*$x[0]-6)**2+
     10*(1-1/(8*$PI))*cos($x[0])+10;

unshift @x, $n;
push @x, $y;

open OT, ">space.temp";
print OT join(" ",@x), "\n";
close OT;
```

objective function

Results

- 21 starting points from a Maximin Latin hypercube design
- SPACE added 12 points prior to terminating
- $\boldsymbol{z}^\star = (3.14042, 2.27273), y^{min} = y(\boldsymbol{z}^\star) = 0.39790$





- **global search**, **local search**

Modifications and Extensions

• EI algorithms can be run in stages, where each stage involves adding multiple input sites. Correlation parameter re-estimation occurs between stages.

• There is no unique improvement criterion for any given class of problems. For example, $I_n(\boldsymbol{z})$ can be modified to provide the user with more control over the trade-off between global and local search: Let g be a non-negative integer and define $I_m^g(\boldsymbol{z}) = (y^{min} - Y(\boldsymbol{z}))^g$ for $Y(\boldsymbol{z}) < y^{min}$ and 0 otherwise. **Larger** values of g correspond to a more **global** search.

• There are no proofs (except in trivial cases) that EI algorithms converge to global optima, but there are many case-studies where they have performed well when compared to other Global Optimization algorithms.

• The EI algorithm has been extended in numerous ways to accommodate situations where (1) there are diffference types of inputs $\boldsymbol{z}$, (2) there are **multiple, competing** outputs, and (3) the output $y(\boldsymbol{z})$ is clearly not stationary (Lee et al., SAMSI 2007)

Examples of Extensions

1. Suppose one subset of the inputs are engineering design variables and the remainder are "field" variables to be integrated with respect to a given probability distribution. Minimize $\int y(\boldsymbol{z}) f(\boldsymbol{z}_{subset}) d\boldsymbol{z}_{subset}$ (Williams et al. 2000)

2. Find $\{\boldsymbol{z} : y(\boldsymbol{z}) = \alpha\}$, the set of $\boldsymbol{z}$ which are α-quantiles of $y(\boldsymbol{z})$ (Roy, 2007, PhD Thesis in progress)

3. Constrained optimization of one output subject to bounds on other computer outputs, i.e.,

$$\min y_1(\boldsymbol{z})$$

subject to

$$y_2(\boldsymbol{z}) \leq B$$

(Williams, submitted)

4. Determine the Pareto Frontier of multiple output functions (Bautista, 2007, PhD Thesis in progress)

III. Summary

- "Space-filling" designs can be used as a tool to explore input space of a computer code as part of the **verfication** process, for screening when "effect sparcity" holds, and are the basis for producing initial estimators of **unknown** process model parameters

- "Process-based" designs – used to improving **overall** prediction, for **optimization** of computer output, for **determining** level set of an output function, for **constrained optimization** of multiple computer outputs, for determining the **Pareto Frontier** of multiple computer outputs, . . .

Questions?