Agenda

1. Generation of Uniform Random Numbers/Vectors with Application to Monte Carlo Simulations
2. Quasi-Monte Carlo Methods
3. Applications in Computer Experiments
4. Summary
Part 1:

Generation of Uniform Random Numbers with Application to Monte Carlo Simulations
When Need the Generation of Uniform Random Numbers/Vectors?

Generation of uniform random numbers/vectors is important in many fields such as:

1. Experimental design
2. Reliability
3. Random number generation (e.g. exponential, normal, and other r.v.s)
4. Sampling
5. Optimization problems
6. Monte Carlo Simulation
7. …
Consider an arbitrary integral over a finite domain $D \subset \mathbb{R}^K$:

$$
\theta = \int_D f(x) \, dx = v(D) \cdot \int_D f(x) \cdot \frac{1}{v(D)} \, dx
$$

$$
= v(D) \cdot E[f(X)],
$$

where $X \sim \text{Uniform}(D)$.

By LLN, the integral can be approximated by the average of the function $f$ at a set of random points $x_1, \ldots, x_n$ simulated from $\text{Uniform}(D)$, i.e.,

$$
\theta \approx v(D) \cdot \frac{1}{n} \sum_{i=1}^n f(x_i) \quad \text{when } n \text{ is large}.
$$
How to Generate Uniform Random Numbers/Vectors?

We will mainly discuss how to generate uniform random numbers/vectors over:

- Hyper-rectangles
- Convex polyhedrons, simplex with restrictions
- Balls and Spheres
- General polyhedrons
- Irregular Domains (often relies on heuristics)
Generation of Uniform Random Numbers (or Vectors) over Hyper-rectangles

- Generation of uniform random numbers over an interval \((a, b)\) is a fundamental task and often relies on the Linear Congruential Methods:
  \[
  X_{n+1} = (aX_n + c) \mod m
  \]
- The algorithms “mt19937ar” and “dsfmt19937” in MATLAB provide a large period approximately \((2^{19937} - 1)\).
- To generate uniform random vectors over a \(K\)-dimensional hyper-rectangle \((a_1, b_1) \times \ldots \times (a_K, b_K)\), one can sequentially generate \(U_1 \sim \text{Uniform}(a_1, b_1), \ldots, U_K \sim \text{Uniform}(a_K, b_K)\) and return the vector \((U_1, \ldots, U_K)\).
Generation of Uniform Random Vectors over $n$-Balls and $n$-Spheres

Let $X = (X_1, \ldots, X_n)$ be a uniform random vector over an $n$-ball centered at 0 with radius $r_0$.

For the distance $R$ from 0, we have:

$$F(r) = P(R \leq r) = \left( \frac{r}{r_0} \right)^n$$

Thus, we know that: $F(R) = \left( \frac{R}{r_0} \right)^n = U \sim \text{Uniform}(0, 1)$

$\Rightarrow R = r_0 \cdot U^{1/n}$
Generation of Uniform Random Vectors over $n$-Balls and $n$-Spheres

We also know that (by Box-Muller Transform) if $Z_1, \ldots, Z_n$ are i.i.d from $N(0, 1)$, then

$$\frac{1}{\sqrt{Z_1^2 + \cdots + Z_n^2}} \,(Z_1,\ldots,Z_n)$$

is uniformly distributed over the unit sphere ($r_0 = 1$). Therefore,

$$X = \frac{R}{\sqrt{Z_1^2 + \cdots + Z_n^2}} \,(Z_1,\ldots,Z_n) = \frac{r_0 \cdot U^{1/n}}{\sqrt{Z_1^2 + \cdots + Z_n^2}} \,(Z_1,\ldots,Z_n)$$
Let \( S \) be a convex polyhedron in the \( n \)-dimensional space having \((k + 1)\) vertices denoted by \( \mathbf{b}_j = (b_{j1}, \ldots, b_{jn})' \), \( j = 1, 2, \ldots, k + 1 \). Let \( \mathbf{B} = (\mathbf{b}_1 - \mathbf{b}_{k+1}, \ldots, \mathbf{b}_k - \mathbf{b}_{k+1}) \) be an \( n \times k \) matrix. Then,

\[
S = \{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{Bz} + \mathbf{b}_{k+1} \},
\]

where \( z = (z_1, \ldots, z_k)' \), \( z_i > 0 \) for \( i = 1, \ldots, k \), and \( \sum_{i=1}^{k} z_i \leq 1 \).

- If \( k \leq n \) and \( \mathbf{x} = (x_1, \ldots, x_n)' \sim \text{Uniform}(S) \), then

\[
\mathbf{x} \overset{d}{=} \mathbf{Bz} + \mathbf{b}_{k+1},
\]

where \( z \) has the \emph{Dirichlet}(1, \ldots, 1; 1) distribution.

(Simplex-Method by Fang et al., 1999)
• If \( k > n \), then \( S \) can be partitioned into \((k - n + 1)\) disjoint \( n \)-simplexes, say, \( S_1, \ldots, S_{k-n+1} \).

Denote the vertices of each \( S_i \) by \( b^i_j = (b^i_{j1}, \ldots, b^i_{jn})', \ j = 1, \ldots, n+1 \), and define the \( n \times n \) matrix \( B^i = (b^i_1 - b_{n+1}, \ldots, b^i_n - b_{n+1}) \), where \( b_{n+1} = b^i_{n+1} \) is a selected common vertex for all of \( S_i \).

Thus, if \( x = (x_1, \ldots, x_n)' \sim Uniform(S) \), then

\[
x \overset{d}{=} \sum_{i=1}^{k-n+1} (B^i z + b_{n+1})I_i(x),
\]

where \( z \) has the \( Dirichlet(1, \ldots, 1; 1) \) distribution, and \( I_i(x) = 1 \) if \( x \in S_i \), and \( I_i(x) = 0 \) otherwise.
Generation of Uniform Random Vectors Over General Polyhedrons

• A general polyhedron (including convex and non-convex polyhedrons) can be represented as the union of a finite number of simplexes.

• Generation of uniform random points over a polyhedron has exact applications in:
  • Sampling
  • Monte Carlo Simulation

• Rejection methods are straightforward but may be inefficient due to low acceptance rates and high computational costs.

• Can extend the idea of Simplex method by Fang et al. (1999).
The above results directly introduce:

**Corollary 1.** Let $G$ be a general $n$-dimensional polyhedron having $k + 1$ vertices and assume that $k > n$. Based on the $k + 1$ vertices, it is clear that $G$ can be partitioned into $m$ disjoint $n$-simplexes, say, $S_1, \ldots, S_m$, where $m \leq k - n + 1$. Denote the vertices of each $S_i$ by $b_j^i = (b_{j1}^i, \ldots, b_{jn}^i)'$, $j = 1, \ldots, n + 1$, and the $n \times n$ matrix $B^i = (b_1^i - b_{n+1}^i, \ldots, b_n^i - b_{n+1}^i)$, $i = 1, \ldots, m$. If $x = (x_1, \ldots, x_n)' \sim \text{Uniform}(G)$, then we have

$$x \overset{d}{=} \sum_{i=1}^{m} (B^iz + b_{n+1}^i)I_i(x),$$

where $z$ has the Dirichlet$(1, \ldots, 1; 1)$ distribution, and $I_i(\cdot)$ is an indicator function such that $I_i(x) = 1$ if $x \in S_i$, and $I_i(x) = 0$ otherwise.
Summary

• Corollary 1 indicates the generation of uniform distribution over convex and non-convex types of polyhedron can be done by simulating the mixture of Dirichlet (1, ⋅⋅⋅, 1; 1) random vectors, while the weights $w_i = v(S_i)/v(S)$ can be simply computed, say, by the Qhull library (http://www.qhull.org).

• The simulation of Dirichlet(1, ⋅⋅⋅, 1; 1) random vectors can be done by using the R package “rBeta2009” (Cheng et al., 2012).

• Since there might be different ways of partitioning $S$ into $m$ simplexes, a rule of thumb is to choose the minimum number of $m$ so as to retain the maximum generation efficiency.

• The same idea works for other types of irregular domains that can be fairly well approximated by polyhedrons.
Example 1

Figure 1: Two possible designated partitions of a non-convex polyhedron with $n = 2$ and $k = 5$. 
Example 2

Figure 2: One possible partition for a non-convex polyhedron. Note that the maximum # of partitions is \((k - n + 1) = 11 - 2 + 1 = 10\).
Applications in Monte Carlo Simulations

Generation of uniform random points over a polyhedron has extensive applications in Monte Carlo Simulation.

Here we introduce two interesting ones:

1. Estimation of Tail Probabilities for the Sum of Random Variables

2. Estimation of Critical Values (or $P$-values) for the Multivariate Kolmogorov-Smirnov Test
Estimation of Tail Probabilities for the Sum of Random Variables

Suppose $X_1, X_2, \ldots, X_n$ are positive random variables with the joint pdf $h(x_1, \ldots, x_n)$, and one would like to evaluate:

$$P\left(\sum_{i=1}^{n} X_i \leq u\right) \quad \text{or} \quad P\left(\sum_{i=1}^{n} X_i > u\right)$$

for a given $u > 0$.

Problems in practice:

(1) The exact distribution of $\sum_{i=1}^{n} X_i$ may be hard to obtain (due to complicated correlation structures) so that one must rely on Monte Carlo simulation.

(2) Direct simulation of the desired multivariate distribution may not be feasible.
Estimation of Tail Probabilities for the Sum of Random Variables

Suppose the random vector \( \mathbf{x} = (x_1, \ldots, x_n)' \) is defined on an open set \( U \subseteq \mathbb{R}^n_+ \) and we consider the \( n \)-simplex 
\[
S = \{ (x_1, \ldots, x_n)' : 0 < \sum_{i=1}^{n} x_i \leq u \} \quad \text{such that} \quad A = S \cap U
\]

is a polyhedron in \( \mathbb{R}^n_+ \).

Define also \( I_A(\mathbf{x}) = 1 \) if \( \mathbf{x} \in A \), and \( I_A(\mathbf{x}) = 0 \) otherwise.

The concept of importance sampling gives

\[
\Pr \left( \sum_{i=1}^{n} X_i \leq u \right) = E_h[I_A]
\]

\[
= \int \cdots \int_{A} I_A(\mathbf{x}) \frac{h(x_1, \ldots, x_n)}{g(x_1, \ldots, x_n)} \cdot g(x_1, \ldots, x_n) dx_1 \cdots dx_n
\]

\[
= E_g \left[ I_A \frac{h(X_1, \ldots, X_n)}{g(X_1, \ldots, X_n)} \right],
\]
Estimation of Tail Probabilities for the Sum of Random Variables

Note that \( g(x_1, \ldots, x_n) \) is an appropriately chosen pdf defined on \( A \).

Since \( A \) is a polyhedron, it is straightforward to choose

\[ g(x_1, \ldots, x_n) = 1/v(A), \]

i.e., the pdf of the uniform distribution over \( A \).

Since \( g(x_1, \ldots, x_n) \) is easy to simulate by the previous result, the desired tail probability can be estimated by the average of

\[ \frac{h(x_1, \ldots, x_n)}{g(x_1, \ldots, x_n)} \]

based on a large number of random samples.
Example: Bivariate Gamma Distribution

Suppose the random vector \((X, Y)\) has McKay’s bivariate gamma distribution (McKay, 1934; Kotz et al., 2000) with pdf

\[
h(x, y) = \frac{r^{a+b}}{\Gamma(a)\Gamma(b)}x^{a-1}(y - x)^{b-1}e^{-ry},
\]

where \(0 < x < y, r > 0, a > 0,\) and \(b > 0.\)

Application in hydrology: \(X =\) drought duration, \(Y =\) subsequent non-drought duration, the \(X + Y\) represents the inter-arrival time of drought events.

Q: How to evaluate \(\Pr(X + Y \leq u)\) ?
Example: Bivariate Gamma Distribution

Let us choose \( r = 1, \ a = b = 2 \) in the bivariate gamma distribution.

Table 1. The estimated tail probabilities for various choices of \( u \) and the computer times (in seconds). Here \( A = \{(x, y)' : 0 < x + y \leq u, \ 0 < x < y\} \), and \( g(x, y) = (1/v(A))I_A(x, y) = (4/u^2)I_A(x, y) \), which is simulated by the R package “rBeta2009”.

<table>
<thead>
<tr>
<th>Value of ( u )</th>
<th>( \hat{\Pr}(X + Y \leq u) )</th>
<th>Computer Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( 10^5 ) points</td>
<td>( 3 \times 10^5 ) points</td>
</tr>
<tr>
<td>( u = 2 )</td>
<td>0.053</td>
<td>0.053</td>
</tr>
<tr>
<td>( u = 4 )</td>
<td>0.294</td>
<td>0.294</td>
</tr>
<tr>
<td>( u = 6 )</td>
<td>0.574</td>
<td>0.574</td>
</tr>
<tr>
<td>( u = 8 )</td>
<td>0.773</td>
<td>0.775</td>
</tr>
<tr>
<td>( u = 10 )</td>
<td>0.889</td>
<td>0.891</td>
</tr>
</tbody>
</table>
The Univariate KS Test

The Kolmogorov-Smirnov (KS) test is used to validate:

$$H_0 : X_1, \ldots, X_n \sim F(x), \ F \text{ can be continuous or discrete.}$$

- The **empirical distribution** of $X_1, \ldots, X_n$ is defined as

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{[-\infty, x]}(X_i),$$

while the **test statistic** is given by

$$D_n = \sup_x |F_n(x) - F(x)|$$

⇒ **Reject $H_0$ when $D_n$ is too large!**
Multivariate KS Tests

Since traditional goodness-of-fit tests (such as KS and Cramér-von Mises tests) do not readily extend to multivariate case, a procedure based on the construction of statistically equivalent blocks (SEB) can be used (see Tukey, 1947; Fraser 1951; Foutz, 1980; Alam et al., 1993).

Let $x_1, \ldots, x_n$ be $n$ observations of the random vector $X$ generated from an arbitrary $N$-variate distribution $F$ having support $S$.

The idea is to first choose $n$ cutting functions $\phi_1, \ldots, \phi_n$ so that $\phi_i(X)$ is continuous for all $i$ and then the support $S$ can be partitioned into a class of “exclusive blocks” $B_1, B_2, \ldots, B_{n+1}$. 
Construction of SEBs

Let \( x_{(1)} = \arg \min_{x \in \{x_1, \ldots, x_n\}} \phi_1(x) \), it is clear that \( S \) is cut into two blocks:

\[
B_1 = \{x \in S : \phi_1(x) \leq \phi_1(x_{(1)})\} \quad \text{and} \quad B_{2\ldots n} = S \setminus B_1.
\]

Next, let \( x_{(2)} = \arg \min_{x \in \{x_1, \ldots, x_n\} \setminus \{x_{(1)}\}} \phi_2(x) \), so \( B_{2\ldots n} \) is cut into two subblocks:

\[
B_2 = \{x \in B_{2\ldots n} : \phi_2(x) \leq \phi_2(x_{(2)})\} \quad \text{and} \quad B_{3\ldots n} = B_{2\ldots n} \setminus B_2.
\]

Continue in this fashion, we see that there exist \( x_{(1)}, \ldots, x_{(n)} \) and \( (n + 1) \) exclusive blocks \( B_1, B_2, \ldots, B_{n+1} \) such that

\[
\bigcup_{i=1}^{n+1} B_i = S.
\]
An Illustration of Constructing the SEBs

Figure 3: The cutting functions introduced by Tukey (1947).
The KS Statistic for Multivariate Distributions

Let $d_i = P_F(X \in B_i)$, Anderson (1966) showed that under $H_0$ \((d_1, \ldots, d_n)\) has a uniform distribution over a unit \(n\)-simplex.

A multivariate version of the KS statistic is defined as:

$$
\tilde{D}_n = \max \left\{ 0, \max_{1 \leq j \leq n} \left( \frac{j}{n} - D_j \right), \max_{1 \leq j \leq n} \left( D_j - \frac{j-1}{n} \right) \right\}
$$

Where $D_j = P_F(X \in \bigcup_{i=1}^{j} B_i) = \sum_{i=1}^{j} d_j$, $j = 1, \ldots, n$.

$\Rightarrow$ When $\tilde{D}_n$ is large, we reject $H_0$: $X_1, \ldots, X_n \sim F$. 
Estimating the Critical Values of the KS Test

To obtain the critical value of $\tilde{D}_n$, one can simulate a large number of random vectors $(d_1, \ldots, d_n)$ from the uniform distribution over the unit $n$-simplex and then estimate the desired percentile of all the computed $\tilde{D}_n$.

For example, to remove the effect of sample size, one can find the value of $d$ such that

\[
\Pr(\sqrt{n}\tilde{D}_n \leq d) \approx 0.95.
\]

Remark: The following formula (Serfling, 1980) can be used to obtain the critical value when $n$ is large:

\[
\lim_{n \to \infty} \Pr(\sqrt{n}\tilde{D}_n \leq d) = 1 - 2 \sum_{j=1}^{\infty} (-1)^{j+1} e^{-2j^2d^2}, \quad d > 0.
\]
Estimating the Critical Values of the KS Test - Example

Table 2. The estimated critical values $\tilde{d}$ of the test statistic $\sqrt{n}\tilde{D}_n$ (given $\alpha = 0.05$) for various sample sizes $n$ and the computed probabilities $\Pr(\sqrt{n}\tilde{D}_n \leq \tilde{d})$ by using Serfling’s formula based on 2,000 Monte Carlo simulations.

<table>
<thead>
<tr>
<th>Sample size $n$</th>
<th>Estimated critical value of $\sqrt{n}\tilde{D}_n$ $\tilde{d}$</th>
<th>$\Pr(\sqrt{n}\tilde{D}_n \leq \tilde{d})$ by Serfling’s formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.3295</td>
<td>0.9417</td>
</tr>
<tr>
<td>100</td>
<td>1.3420</td>
<td>0.9455</td>
</tr>
<tr>
<td>200</td>
<td>1.3501</td>
<td>0.9478</td>
</tr>
<tr>
<td>300</td>
<td>1.3567</td>
<td>0.9496</td>
</tr>
<tr>
<td>400</td>
<td>1.3641</td>
<td>0.9509</td>
</tr>
<tr>
<td>500</td>
<td>1.3578</td>
<td>0.9499</td>
</tr>
</tbody>
</table>

better
Summary

- Monte Carlo Simulation method is attractive since it requires less sophisticated math.

- If MC simulation is used for evaluating the integration (e.g., any expectation can be written as an integral), the convergence rate is $O(n^{-1/2})$ (in probability).

  **Advantages**: Independent of dimensionality and robust

  **Shortcoming**: The price of robustness and independence is low convergence rate due to “clumping” (or non-uniformity), especially for high-dimensional problems.

- An alternative is **Quasi-Monte Carlo method**, which is a deterministic approach to mimic pseudo-random sequence.
Part 2:

Quasi-Monte Carlo Methods
Monte Carlo vs Quasi-Monte Carlo Methods

The difference between MC and Quasi-MC methods is the way the points $x_1, \ldots, x_n$ are chosen.

MC: $x_1, \ldots, x_n$ are from a pseudo-random sequence

Quasi-MC: $x_1, \ldots, x_n$ are found so as to maintain the maximum uniformity and result in a possible faster convergence rate (e.g., closer to $O(n^{-1})$)

Therefore, it is necessary to define a good measure of “uniformity”.
An Illustration: MC vs Quasi-MC

MC

Quasi-MC
Discrepancy –
A Measure of Uniformity

Let $P = \{x_1, \ldots, x_n\}$ be a set of $n$ points in a convex domain $D \subseteq \mathbb{R}^K$.

The “discrepancy” (or star discrepancy) is defined as

$$D(n,P) = \sup_{x \in D} \left| \frac{N(x,P)}{n} - \frac{\nu([0,x))}{\nu(D)} \right|,$$

where $[0, x)$ is the rectangular region from the origin and $N(x, P)$ is the number of points inside the rectangular region.

The best allocation of $P$ is:

$$P^* = \arg\min_P D(n,P)$$
An example of one allocation $P$ with $n = 12$ (points) in a square $C^2 = [0, 1] \times [0, 1]$. 

\[
\frac{N(x, P)}{n} = \frac{2}{12}
\]
Some Variants of Discrepancy Measures

There are various measures of uniformity based on the concept of “discrepancy”:

- *F*-discrepancy (Wang and Fang, 1990): can incorporate “weights” into the formulation.
- (Star) $L_p$-discrepancy (Weyl, 1916): taking the overall $L_p$-norm discrepancy instead of $L_1$-norm maxima.
- Symmetrical discrepancy (Ma, 1997): solving rotational problems
- Centered $L_2$-discrepancy (Hickernell, 1998): solving rotational problems and incorporating uniformity in lower dimensions
(Star) $L_p$-Discrepancy

The (star) $L_p$-discrepancy is defined as (Weyl, 1916):

$$L_p(n,P) = \left\{ \int_{C^s} \left| \frac{N(x,P)}{n} - \nu([0,x]) \right|^p dx \right\}^{1/p} ,$$

where instead of taking the “supreme” of the difference between $N(x, P)/n$ and $\nu([0, x])$, the overall $L_p$-norm difference is considered.
Rotational Problems
Symmetrical Discrepancy

In order to overcome the rotational problem, Ma (1997) proposed the measure of symmetrical discrepancy, which is defined as:

\[ SD(n, P) = \frac{1}{2^K} \sum_{i=1}^{2^K} D(n, P_i), \]

where \( P_i \) represent the allocation of \( n \) points with the origin 0 assigned to the \( i \)-th vertex of the hypercube \([0, 1]^K\).

For example, if \( K = 2 \), there are \( 2^2 = 4 \) star discrepancies to be considered (see the previous slide).
Centered $L_p$-Discrepancy

The Centered $L_p$-discrepancy (Hickernell, 1998) is defined as:

$$CL_p(n,P) = \left\{ \sum_{u \neq \phi} \int_{C^u} \left| \frac{1}{n} N(J_{x_u}, P_u) - \nu(J_{x_u}) \right|^p dx_u \right\}^{1/p},$$

where $u$ is a subset of $\{1, \ldots, K\}$, $x_u$ is a point in the projected $u$-dimensional space, and

$$J_{x_u} = \{ x \in C^u : \min(a_j, x_j) \leq x < \max(a_j, x_j) \text{ for } 1 \leq j \leq u \}.$$
Centered $L_\rho$-Discrepancy: A 2-D illustration

Projection in the subspace $C_u = C^2$
Some Remarks on the CL$_p$

- The Centered $L_p$-discrepancy solves the “rotational problem” by choosing the origin as the closest vertex to a point $x$.

- The Centered $L_p$-discrepancy also takes into account the uniformity in the “lower dimensional spaces”.

- When $p = 2$, it has connections to orthogonality, minimum abbreviation, and confounding for a class of designs.
Central Composite Discrepancy (CCD)

Note that All the above-mentioned methods apply to regular domains (hypercubes, hyperrectangles, or regions obtained by restricted transformations).

Chuang and Hung (2010) proposed a new measure called Central Composite Discrepancy (CCD), which has the following sophisticated properties:

(1) Applicable to any irregular domains
(2) Solution is invariant under coordinate rotation
(3) Lower dimensional uniformity can be incorporated
(4) Idea of weights can be incorporated
Steps of Constructing the CCD

- For each point \( x = (x_1, \ldots, x_K) \in D \), denote
  \[
  x_k^{(0)} = \{ r \in R : r \leq x_k \} \quad \text{and} \quad x_k^{(1)} = \{ r \in R : r > x_k \}
  \]
  for each dimension \( k, k = 1, \ldots, K \).

- Thus, the input region \( D \) is decomposed into \( 2^K \) subregions, where the \( k \)-th subregion is denoted by
  \[
  D_k(x) = \{ x_1^{(i_1)} \times \cdots \times x_K^{(i_K)} \} \cap D,
  \]
  and \( (i_1, \ldots, i_K) \) is the binary expression of \( K - 1 \).
Illustration of a 2-D Partition
Formulation of the CCD

For a set of $n$ points $P = \{x_1, \ldots, x_n\}$ and any point $x$ in $D$, let us define

$$N(D_k(x), P) = \sum_{i=1}^{n} \mathbf{1}\{x_i \in D_k(x)\}.$$

The Central Composite Discrepancy (CCD) is defined as

$$CCD_p(n, P) = \left\{ \frac{1}{v(D)} \int_D \frac{1}{2^K} \sum_{k=1}^{2^K} \left| \frac{N(D_k(x), P)}{n} - \frac{v(D_k(x))}{v(D)} \right|^p \right\}^{1/p}$$

where $p > 0$, and $v(\cdot)$ denotes the volume.
The CCD with Weights

Let $f(x)$ be a nonnegative function defined on $D$ and such that $\int_D f(x) dx = 1$.

Q: How to find $n$ points $P = \{x_1, \ldots, x_n\}$ on $D$ so that they have the “best representation” for the function $f(x)$?

⇒ The solution is the minimizer of:

$$WCCD_{f,p}(n, P) = \left\{ \frac{1}{v(D)} \int_D \frac{1}{2^K} \sum_{k=1}^{2^K} \frac{N(D_k(x), P)}{n} - F(D_k(x)) \right\}^{1/p}$$

where $F(D_k(x)) = \int_{D_k(x)} f(x) dx$
Lower Dimensional Uniformity

To take into account the lower dimensional uniformity, we defined the “projected CCD” as:

\[
P_{\text{CCD}}_{\psi, f, p}(n, P) = \left\{ \frac{1}{|\Psi|} \sum_{\psi \in \Psi} W_{\text{CCD}}_{f_{\psi}, p}(n, P_{\psi}) \right\}^{1/p},
\]

where \( \Psi \) is the collection of all subsets of \( \{1, \ldots, K\} \), \( \psi \in \Psi \),
\[
f_{\psi} = \int f(x) dx_{-\psi}, \quad dx_{-\psi} = \prod_{i \in \{1, \ldots, K\} \setminus \psi} dx_i,
\]
and \( P_{\psi} \) is the projection to the subspace \( \psi \).
Summary

Compared to the MC method, the quasi-MC method has the following features:

**Advantages:**

- Faster convergence (or more accurate) due to the property of “uniformity”. The worst case convergence rate \( = O(n^{-1}(\log n)^K) \), \( K = \) some constant \( \Rightarrow \) can be very close to \( O(n^{-1}) \).

**Disadvantages:**

- Loss of “randomness” (but on the other hand suitable for applications such as computer experiments)
- May encounter serious computational problems in high-dimensional spaces (e.g., finding minimum CCD if \( K \) is large).
The quasi-MC method has received great attention in the area of computer experiments – it is also known as the “Uniform Design” (Fang, 1978, 1980).

The following are some extensive applications:

• Fitting parametric/nonparametric response models (robust to model selection)
• Detecting the target region or exploring the relationship between the responses and variables in computer experiments.
• Reliability testing
• Product design in manufacturing
• Solving complex optimization problems with linear/nonlinear constraints.
Construction of Nearly Uniform Designs (NUD)

• Finding the optimal solution $P^*$ based on CCD in a continuous domain is an **NP-hard** problem.

• In practice, one usually considers a **discrete version** by superimposing grids over the domain $D$. The solution based on the grids is called the **Nearly Uniform Design (NUD)**.

• Let’s consider a $K$-dimensional rectangular domain with the number of grids $N = m^K$ (i.e., $m$ grids for each dimension).

Let $n$ be the number of experimental runs (# of grids to be selected) and usually (but not necessary) we assume:

$$K < n \leq m.$$
A Discrete Approximation of CCD

Let $G$ be a set of $N = m^K$ grid points that represents the input domain $D$. The following quantity is a discrete approximation of CCD:

$$\tilde{CCD}_p(n,P) = \left\{ \frac{1}{|G|} \sum_{g \in G} \frac{1}{2^K} \sum_{k=1}^{2^K} \left[ \frac{N(D_k(g),P)}{n} - \frac{v(D_k(g))}{v(D)} \right]^p \right\}^{1/p}$$

The NUD is the allocation $P^*$ that minimizes the above quantity.
Computational Challenges of Constructing the NUD

Even for a fairly small number of $K$ and $n$, finding the NUD based on the grids can be computationally intricate !

**Algorithm 1:** (Exhaustive/Greedy Search)

Assume $K < n \ll N$, the total number of computation time is

$$\binom{N}{n} = O(N^n).$$

**Example:**

If $K = 3$, $n = 6$, $N = 10^3$, the order of computation time is $10^{18}$. 
Computational Challenges of Constructing the NUD

Algorithm 2: (Switching algorithm by Chuang and Hung, 2010)

Idea: Place an initial design $P^{(0)}$, switch one grid point in $P^{(0)}$ with one grid outside if the CCD can be reduced. Repeat and stop until CCD can not be further reduced.

Assume $K < n << N$, the upper bound of computation time is

$$N^{(1+p)}2^K n(N - n) = O(N^{2+p}).$$

Example: If $K = 3$, $n = 6$, $N = 10^3$ and $p = 2$ ($L_2$-norm is a popular choice in CCD), the maximum computation time $\approx 10^{12}$. 
Other Algorithms

• Threshold Accepting Algorithm (Lin et al., 2010, CSDA)
• Particle Swarm Optimization (Chen et al., 2015, CSDA)

⇒ All these studies intend to obtain a design that possibly minimizes the CCD, while the computation time of these two methods was not valid !!

• Since researchers are more interested in the high-dimensional cases (say, $K$ is large while $n$ is not particularly large), there is indeed a need of more efficient algorithms.
Hung (2017) proposed an accelerated search procedure that can be applied to many existing algorithms:

**Algorithm of the Accelerator**

**Initial Setup:** Select $\psi_0 \subset \{1, \ldots, K\}$, find the optimal design $\mathcal{P}_0^*$ in the selected $|\psi_0|$-dimensional space by using a selected search method. Set $i = 0$.

**Step 1:** Select $\psi_{i+1} \subset \{1, \ldots, K\} \setminus \psi_i$, set $\psi_{i+1} = \psi_{i+1} \cup \psi_i$.

**Step 2:** Find the optimal design $\mathcal{P}_{i+1}^*$ in $D_{\psi_{i+1}}$ by augmenting the design $\mathcal{P}_i^*$ in the lower dimensional input domain $D_{\psi_i}$.

**Step 3:** If $|\psi_{i+1}| = K$, then stop and extract $\mathcal{P}_{i+1}^*$; otherwise set $i = i + 1$, go back to Step 1.
The following theorem shows the computational performance of the proposed Accelerator.

**Theorem:** If the number of superimposed grids in each dimension of $D$ is $N^\frac{1}{K}$ and $|\psi_i| = d(i + 1)$ for each iteration $i$ (where $d$ is divisible by $K$), the “accelerated exhaustive search” has the computation time $O(N^{\frac{nd}{K}})$, while the “accelerated Switch Algorithm” has the computation time $O(N^{\frac{d}{K}(2+p)})$.
Recall the early example of $K = 3$, $n = 6$, $N = 10^3$, $p = 2$, and now let us assume $d = 1$.

The following are the computation times for finding the NUD:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Without Acceleration</th>
<th>With Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exhaustive Search</td>
<td>$10^{18}$</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Switch Algorithm</td>
<td>$10^{12}$</td>
<td>$10^4$</td>
</tr>
</tbody>
</table>
Uniformity of the Search Algorithms w/o Acceleration

**Area:** $[0, 1] \times [0, 1]$

**# of Grids:** $N = 10^2$

![Graph showing estimated CCD² vs. number of design points for Exhausted, Switching, Acc. Exhausted, and Acc. Switching methods.]
Summary

• What’s the performance of the proposed acceleration search procedure when the input dimension $K$ is high?
• Given the same experimental budget (i.e., the same number of design points $n$), which search algorithm performs best (most accurate or having the fastest convergence rate to the desired integral, not to the proposed discrepancy)? Especially when $K$ is large, and $n$ is not large.
• How does the result change for different choices of $p$ (norm of distance), $m$ (grid level), shapes of the domain $D$, or forms of the integral? (i.e. robust to the change of parameters)
• For high-dimensional domains, how to integrate the idea of Dimensional Analysis (DA) with UD?
Part 3:

Applications in Computer Experiments
What are Computer Experiments?

• A typical computer experiment simulates complex physical or engineering systems by using a computer code. However, simulation of such computer code is often computationally expensive.

• The deterministic computer code yields identical output (or response) \(\Rightarrow\) Lack of randomness!

• To explore the response surface of interest, one often relies on the Bayesian framework (assume a prior structural model).
What Questions we are interested in?

• How to explore the relationship between the input factors and the response measure of interest? (modeling issue)

• What are the settings of the input factors so as to produce the maximum/minimum response? (optimization problem)

• What are the settings of the input factors so as to produce a designated response (e.g. contour of a particular height, infinite outputs, jump discontinuities, etc).

• Multiple response problems, etc

⇒ The issue of modeling and design of computer experiments!
Traditional Methods for Solving Optimization Problems

- Interchange Arguments
- Forward/Backward Induction
- Stochastic Optimization (e.g. Markovian assumptions)
- Linear/Quadratic/Mixed-Integer/Nonlinear Programming (there exist a bunch of algorithms)
- Large Deviation and Approximations (e.g. diffusion approximation)
Some Challenges

Traditional methods can not simply handle the optimization problems due to:

- Complex relationships between inputs and responses
- Invalid analytical results (e.g. weak assumptions)
- High computational expenses (an important feature of computer experiments)
Response Surface Methodology (RSM) -- A Classical Approach

- Box and Drapper, 1987.
- An optimal search procedure for searching maximum/minimum involving statistical and mathematical techniques.
- Most applications are in the industrial world, where several input variables $\bar{x} = (x_1, \cdots, x_k)$ influence some performance measure $f(\bar{x})$ (i.e. response) of interest.
- Suitable when the underlying response surface is smooth (i.e. with continuous $f(\bar{x})$ and $\bar{x}$).
Steps of RSM

- **Step 1**: Pick an initial input point \( \bar{x} \) as a center and consider a \( 2^k \) factorial design around it.

- **Step 2**: Derive the responses for the input points selected in Step 1 and fit a second order model. Check out the goodness of fit using normal Regression Analysis.

- **Step 3**: If the second order model is not adequate, fit a simple linear model. Consider the path of steepest ascent/decent and select the change point to be a new input center, go back to Step 1.

- **Step 4**: If the second order model is adequate, select additional input points using Central Composite Design to improve the fitting of the second order model.

- **Step 5**: Find the input location corresponding to the maximum (or minimum) based on the second order model derived in Step 4.
$2^k$ Factorial Design

$k = 2$

$k = 3$
Central Composite Design (CCD)

(2 input factors)  (3 input factors)
A Graphical Illustration of RSM
Some Remarks

- **Scale choice** of each input factor for design
  - By experience or simulation budget.
- Using **replicates**? (random effect or deterministic output ?)
- **Long iterations between Step 1 to Step 3** indicate that the response surface might be flat or monotone ➔ Check out the **boundary** of the input domain for maximum/minimum.
- Maximum, Minimum, Saddle Point, Ridge System?
- Local or global maximum/minimum?
The Gaussian Stochastic Process (GASP) Model

Denote the input point by $x_i = (x_{i1}, \ldots, x_{iK})$ and its associated output by $y_i = y(x_i)$. The experimental output is modeled as

$$y(x_i) = \mu + z(x_i), \quad i = 1, \ldots, n,$$

where $\mu$ represents the overall mean and $z(x_i)$ is a spatial error term with $E(z(x_i)) = 0$, $\text{Var}(z(x_i)) = \sigma_z^2$, $\text{cov}(z(x_i), z(x_j)) = \sigma_z^2 R_{ij}$, and

$$R_{ij} = \text{corr}(z(x_i), z(x_j)) = \prod_{k=1}^{K} \exp \left\{ -\theta_k |x_{ik} - x_{jk}|^{p_k} \right\} \quad \text{for all } i, j.$$

Note that $\theta = (\theta_1, \ldots, \theta_K)$ is a vector of hyperparameters, while $p_k$ is often chosen as $p_k = 2$. 
The Gaussian Stochastic Process (GASP) Model

In general, we assume

\[ y = (y(X_1), \ldots, y(X_n)) \sim \mathcal{N}_n(1_n \mu, \Sigma), \text{ where } \Sigma = \sigma_z^2 R \text{ and } R = [R_{ij}]. \]

Based on the likelihood (and profile likelihood) approach, the best linear unbiased predictor (BLUP) for the response at any un-tried input point \( x^* \) is given by

\[ \hat{y}(x^*) = \hat{\mu} + r' R^{-1} (y - 1_n \hat{\mu}), \]

with the mean squared error

\[ s^2(x^*) = \sigma_z^2 \left( 1 - r' R^{-1} r + \frac{(1 - 1_n' R^{-1} r)^2}{1_n' R^{-1} 1_n} \right), \]

where \( r = (r_1(x^*), \ldots, r_n(x^*))' \) and \( r_i(x^*) = \text{corr}(z(x^*), z(x_i)) \).
Example: Contour Estimation in Computer Experiments

Suppose we would like to estimate a contour $S(a) = \{ x : y(x) = a \}$ that represents the set of input points with the response value $a$.

Example: The Goldprice function defined on $[0, 1] \times [0, 1]$, where we now choose $a = 1.5 \times 10^5$.

$$f(x_1, x_2) = \left[ 1 + \left( \frac{x_1}{4} + 2 + \frac{x_2}{4} \right)^2 \left\{ 5 - \frac{7x_1}{2} + 3 \left( \frac{x_1}{4} + \frac{1}{2} \right)^2 \right. \right. $$
\[ - \frac{7x_2}{2} + \left( \frac{3x_1}{2} + 3 \right) \left( \frac{x_2}{4} + \frac{1}{2} \right) + 3 \left( \frac{x_2}{4} + \frac{1}{2} \right)^2 \left. \right\} \right] \times \left[ 30 + \left( \frac{x_1}{2} - \frac{1}{2} - \frac{3x_2}{4} \right)^2 \left\{ 26 - 8x_1 + 12 \left( \frac{x_1}{4} + \frac{1}{2} \right)^2 \right. \right. \\
\left. \left. + 12x_2 - (9x_1 + 18) \left( \frac{x_2}{4} + \frac{1}{2} \right) + 27 \left( \frac{x_2}{4} + \frac{1}{2} \right)^2 \right\} \right]. \]
Sequential Design for Contour Estimation

The following procedure was introduced by Ranjan et al. (2008) for estimating $S(a)$ based on GASP:

**Initial Setup**: Determine an appropriate design domain $D$ and the size of initial design $n_1$. Allocate an adequate initial design $P^*_1$, say, by Latin Hypercube Designs (LHDs) or Uniform Design (UD).

**Step 1**: Fit a GASP model based on the current design $P^*_s$, select a next-stage design point $x^*$ which maximizes the expected value of an improvement function, i.e., $E[I(x)]$ (see later).

**Step 2**: Obtain the experimental output at $x^*$ and set $P^*_{s+1} = P^*_s \cup \{x^*\}$, let $s = s + 1$.

**Step 3**: Stop and extract $S(a)$ or go to **Step 1**.
**Improvement Function**

After fitting the GASP model at each design stage $s$, we define the *improvement function* of an untried input point $x$ by:

$$I(x) = \epsilon^2(x) - \min\{(y(x) - a)^2, \epsilon^2(x)\},$$

where $\epsilon(x) = \alpha s(x)$ for some $\alpha > 0$ and $y(x) \sim N(\hat{y}(x), s^2(x))$.

The expected value of $I(x)$ is given by:

$$E[I(x)] = \left[\alpha^2 s^2(x) - (\hat{y}(x) - a)^2 - s^2(x)\right] \cdot [\Phi(u_2) - \Phi(u_1)]$$

$$+ s^2(x) \left[u_2 \phi(u_2) - u_1 \phi(u_1)\right] + 2s(x)(\hat{y}(x) - a) \left[\phi(u_2) - \phi(u_1)\right],$$

where $u_1 = (a - \hat{y}(x) - \alpha s(x))/s(x)$, $u_2 = (a - \hat{y}(x) + \alpha s(x))/s(x)$.
Choosing the Next-stage Design Point

The next design point at the stage $s$ is selected as

$$x^* = \arg \max_{x \notin \text{current design}} E[I(x)]$$

The above rule of choosing $x^*$ involves the tradeoff between the following two concepts:

- Tend to produce an output close to $a$;
- Tend to have a large prediction error.
Demo of Contour Estimation

(a) 

(b) 

(c) 

(d) Final contour and true contour
Some Remarks

- Both theoretical and numerical results show the convergence of the estimated contour line.

- How to choose the best initial design so as to accelerate and improve the estimation accuracy under a limited experimental resource?

- How to deal with the response surface with discontinuities?

- Other modeling techniques? E.g. Stochastic Kriging by Chen and Zhou (2017, EJOR).

- Performance in high-dimensional spaces?


