Sampling-Based Uncertainty Propagation

- We've already discussed simple random sampling
  - And its draw backs
  - $1/\sqrt{N}$

- Now, we'll talk about more sophisticated methods

- A couple of notes for generating samples for simulation-based UQ
  1. Assuming deterministic codes, we do not want to replicate an experiment
  2. The samples should be applicable to several studies if possible.

- The process of generating the points to run the code at is called "design of computer experiments."

- A lot of today's content can be found in Santner, Williams, and Notz "D.A.C.E"

- In general we want space-filling designs because we don't know where in the input space extremes values of outputs will occur

- All of the models we discuss can be used to build emulators

Sampling-Based UQ

Notation we want to know dist of $\underline{Y} = y(\vec{X})$ where $\underline{X} = \vec{x}$ (both $\underline{X}$ and $\underline{Y}$ are R.V.'s)

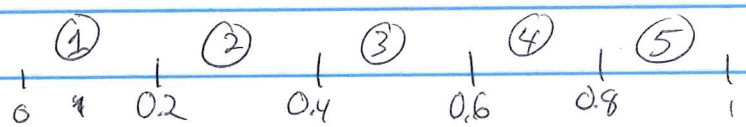
Elementary Methods

- Other than simple sampling, there is a simple strategy that actually works fairly well.

Stratified Sampling

- Consider the uniform distribution $\underline{X} \sim U(0,1)$ if I generate 5 samples from this distribution there is no guarantee that the samples won't be close together

- Instead divide $[0,1]$ into 5 intervals and ~~pick~~ sample each point in one of the intervals



Generalize this to M intervals and write

$$x_j \sim U\left(\ell/M, \ell+1/M\right) \quad \text{where } \ell = j \bmod M$$

i.e. when $j = 0, M, 2M, \ldots$ $x_j \sim U(0, 1/M)$

$j = 1, M+1, 2M+1, \ldots$ $x_j \sim (1/M, 2/M)$

$\vdots$

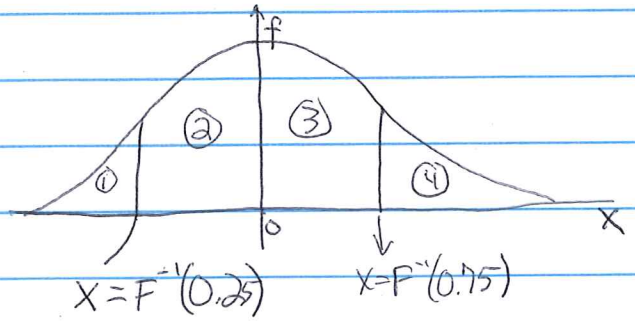
Sampling-Based UQ

- We can build on this to sample from more interesting distributions

- For a general dist the CDF has the property that $F(x) \in [0,1]$

- Therefore, divide the unit interval into M bins, and use the above procedure to sample all around a distribution.

- Example, let $M=4$ and $\overline{X} \sim N(0,1)$



- This will generate samples from the "tails" more often

- Can make this multi-D

2-D Case
M=3
in each direction

3,1	3,2	3,3
2,1	2,2	2,3
1,1	1,2	1,3

Sampling-Based UQ

- Can do this for any distribution

- Latin Hypercube Designs

- A Latin square divides a square into n^2 equally sized cells, then in each row the integers 1-n are placed so that no column has ~~more than one~~ an integer repeating. (Sudoku-like)

n=4 examples

3	1	4	2
4	3	2	1
2	4	1	3
1	2	3	4

4	3	2	1
3	4	1	2
2	1	4	3
1	2	3	4

if I randomly pick an integer from 1 to n, then I draw samples from each box with that number

example

draw a 4

3	1		2
	3	2	1
2		1	3
1	2	3	

	3	2	1
3		1	2
2	1		3
1	2	3	

The boxes I draw my samples from are evenly spread out

Sampling-Based UQ

In general when $\vec{X} = (X_1, X_2, \ldots X_d)$ and X_k are independent we create a Latin Hypercube Design (LHD)

- Suppose we want n samples,
- Divide the domain of each X_k into n intervals
- The Cartesian product of these samples form a set of cells that partition the d-dimensional space
- There are n^d cells

- Specifically, divide X_k into n cells with edges $F_k^{-1}(0), F_k^{-1}\left(\frac{1}{n}\right), F_k^{-1}\left(\frac{2}{n}\right), \ldots F_k^{-1}\left(\frac{n-1}{n}\right), F^{-1}(1)$

- To choose the cells let $\underline{\underline{\Pi}} = (\Pi_{jk})$ be a $n \times d$ matrix where
  - The columns are d, different randomly selected permutations of $\{1, 2, \ldots, n\}$

- To generate the j^{th} sample, $j = 1, \ldots, n$

$$X_{jk} = F_k^{-1}\left(\frac{1}{n}\left(\Pi_{jk} - 1 + U_{jk}\right)\right)$$

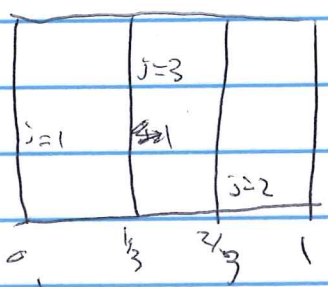
where $U_{jk} \sim U(0,1)$

Sampling-based UQ

Example $\vec{X} = (X_1, X_2)$ with $X_1, X_2 \sim U(0,1)$

Generate a LHD for $n=3$

First generate $\underline{\underline{\Pi}} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 3 \end{pmatrix}$



X_{ijk}

$$X_{1,1} = \frac{1}{3}(U_{11}) \qquad X_{1,2} = \frac{1}{3}(2-1+U_{12})$$

$$X_{2,1} = \frac{1}{3}(U_{2,1}+2) \qquad X_{2,2} = \frac{1}{3}U_{22}$$

$$X_{3,1} = \frac{1}{3}(U_{3,1}+1) \qquad X_{3,2} = \frac{1}{3}(2+U_{3,2})$$

Not all LHD's are equally space-filling

$\underline{\underline{\Pi}} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$ isn't so good.

Orthogonal Arrays

- An orthogonal array O on s symbols of strength t is an $n \times p$ $(p \geq t)$ matrix whose entries are the s symbols so that in every $n \times t$ submatrix of O, every s^t possible rows appear the same number, λ, times.

$n = \lambda s^t$

Sampling-Based UQ

Example for $p=3$ (3-Dim) $n=9$, $s=3$, $t=2$, $\lambda=1$

$n = \lambda s^t = 9 = 3^2$

- An orthogonal array is

$$\begin{pmatrix} 1&1&1 \\ 1&2&2 \\ 1&3&3 \\ 2&1&2 \\ 2&2&3 \\ 2&3&1 \\ 3&1&3 \\ 3&2&1 \\ 3&3&2 \end{pmatrix}$$

each 9×2 array spans the entire 2-D space

- Note LHD are strength 1 Orthogonal arrays
- these can be built sequentially

Properties of Sampling-Based Designs

Consider $Y = g(\vec{X})$ with $F_k(x)$ the CDF for var X_k, $g(\cdot)$ is a real-valued function.

The mean of $g(Y)$ is $\mu = E[g(Y)] = \int g(y(x)) f(x) dx$

- The naive moment estimator is

$$T = \frac{1}{n} \sum_{j=1}^{n} g(y(\vec{X}_j))$$

$\vec{X}_j$ is a random sample

Sampling-Based UQ

$$\sigma^2 = \mathrm{Var}\{g(y)\}$$

For a simple random sample call T, T_R

$$E\{T_R\} = \text{mean of } \frac{1}{n}\sum_j g(y(X_j)) = \mu$$

$$\mathrm{Var}(T_R) = \frac{\sigma^2}{n}$$

- This implies T_R is an unbiased estimator of μ
- For stratified sampling, call T, T_S

$$T_S = \sum_{i=1}^{I}\left[\left(\frac{p_i}{n_i}\right)\sum_{j=1}^{n_i} g(y(\vec{X}_{ij}))\right] \quad \text{where}$$

I number of strata and n_i is the number of samples in each stratum, p_i is portion of sample space in the ith stratum

$$E\{T_S\} = \mu \quad \text{and}$$

$$\mathrm{Var}\{T_S\} = \mathrm{Var}\{T_R\} - \frac{1}{n}\sum_{i=1}^{I} p_i(\mu - \mu_i)^2$$

$$\Rightarrow \mathrm{Var}\{T_S\} < \mathrm{Var}\{T_R\}$$

$n = \sum_{i=1}^{I} n_i$

- The bigger $(\mu - \mu_i)^2$ is, the better

Sampling-based UQ

Now let $g(y) = y$ (identity function)

Also write $F(\vec{x}) = \prod_{i=1}^{d} F_i(x_i)$ (independent inputs)

also $F_{-j}(\vec{x}_{-j}) = \prod_{i=1, i\neq j}^{d} F_i(x_i)$

Let $\mu = \int y(\vec{x}) f(\vec{x}) d\vec{x}$ and $\alpha_j(x_j) = \int [y(\vec{x}) - \mu] f_{-j}(\vec{x}_{-j}) d\vec{x}_{-j}$

- μ is the overall mean and the d α_j's are the "main effects": how does y change on average if I only change one variable.
- Define $r(\vec{x}) = y(\vec{x}) - \mu - \sum_{i=1}^{d} \alpha_i(x_i)$ is the residual of $y(\vec{x})$

Theorem: Let $\mathrm{Var}_{LHS}\{\bar{Y}\}$ = variance of $\bar{Y}$ as estimated by LHS

$\mathrm{Var}_{SRS}\{\bar{Y}\}$ = variance as estimated by simple random sampling

$$\underset{n\to\infty}{\mathrm{Var}_{LHS}}\{\bar{Y}\} = \frac{1}{n}\int r^2(\vec{x}) f(\vec{x}) d\vec{x} + O(n^{-2})$$

$$\mathrm{Var}_{SRS}\{\bar{Y}\} = \frac{1}{n}\int r^2(\vec{x}) f(\vec{x}) d\vec{x} + \frac{1}{n}\sum_{i=1}^{d}\int \alpha_i^2(x_i) f_i(x_i) dx_i + O(n^{-2})$$

Sampling-based UQ

Also $\bar{Y}$ tends to a normal distribution under LHS sampling

Theorem $\sqrt{n}\left(\bar{Y}-\mu\right) \sim N\left(0, \int r^2(\vec{x})d\vec{x}\right)$ as $n\to\infty$

Designs Based on Distance

Consider a distance metric

$$\rho_p(\vec{x}_1, \vec{x}_2) = \left[\sum_{j=1}^{d} |x_{1j} - x_{2j}|^p\right]^{1/p}$$

$p=2$ is Euclidean dist, $p=\infty$ is defined as

$$\rho_\infty(\vec{x}_1, \vec{x}_2) = \max_j |x_{1j} - x_{2j}|$$

- Now we want designs that maximize the minimum distance between points

the average distance is given by

$$m_{(p,\lambda)}(D) = \left(\frac{1}{\binom{n}{2}} \sum_{\vec{x}_i,\vec{x}_j \in D} \left[\frac{d^{1/p}}{\rho_p(\vec{x}_i,\vec{x}_j)}\right]^\lambda\right)^{1/\lambda}$$

For fixed (p,λ), the optimal condition is D_{av}

Therefore, $m_{(p,\lambda)}(D_{av}) = \min_{D\subset X} m_{(p,\lambda)}(D)$

optimal in sense that there is non-redundancy

Sampling-based UQ

The minimax design is

$$m_{(p,\infty)} = \min_{D \subset X} m_{(p,\infty)}(D)$$

The problem here is the fact that there is a large number of possible designs to test

Matlab has a function that can generate a minimax LHD

- That is the LHD that maximizes the minimum distance between points

(Show page 141 of SWN)

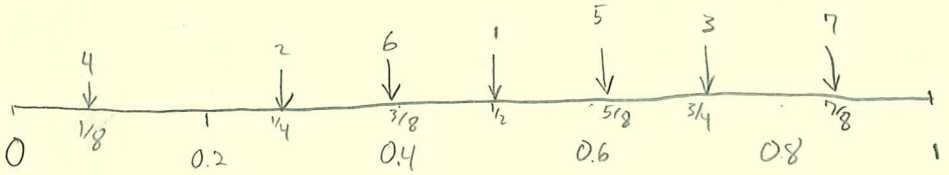
Quasi-Monte Carlo

(Quasi - apparently but not really)
Pseudo - supposed, but not really so

- Use seemingly random sequences that aren't really random
- Low discrepancy sequences - have a measure of uniformity, no large gaps or clustering.
- Idea: instead of using Rand(0,1) we sample from a seemingly random sequence with nice properties

van der Corput Sequence (1935)

- fill in largest gap (Base 2 case)



Procedure:

for n=1.. N

Write n in base b
reflect the number in base b
Convert the reflected base b number to a decimal

end for

Example b=2

n=1 is 01 in base 2

reflect: .1 in base 2 → .1

write as decimal $1\times 2^{-1} = \frac{1}{2}$

n=4 is 100

reflect: 0.001

dec: $1\times 2^{-3} + 0\times 2^{-2} + 0\times 2^{-1} = \frac{1}{8}$

- Now we can generate numbers on the unit interval
- Use these rather than rands to sample
- Show figures of van der Corput normals
- Bases are prime numbers

Multi-D Low Discrepancy Sequences

- Need to avoid multi-D clustering
- Can't just do van der Corput in each dimension (we'll only sample diagonal.

Halton Sequence

- For each increasing dimension increase the base
- For a 2-D samples use base 2, 3
  3-D use base 2, 3, 5
- Have to use prime numbers
- As ~~base~~ D gets large, cycle length gets large
- Halton ~~not~~ good for ~6, 8 dimensions
  - > ~~Coffe~~ Higher base, more numbers it takes to cover [0, 1] and turn around

Faure Sequence

- Like Halton, but uses one base and permutes elements in each dimension
- Computationally slower

Sobol Sequence

- Always use base 2, but complicated reordering
- Not simple: Galois theory used to generate irreducible primative polynomials

Rate of Convergence

- Standard MC converges $O(N^{-1/2}) = \sigma$
- QMC can converge at $O(N^{-1})$ (best case)

  Worst case $O((\ln N)^d / N)$ d = # of dimensions

Conv.

d	N	$1/\sqrt{N}$	$1/N$	$(\ln N)^d/N$
1	1,000	.03162	0.001	0.00691
1	100,000	.00316	0.00001	0.00012
2	10,000	.01	0.0001	0.00848
5	10,000	0.01	0.0001	6.628
10	10,000	0.01	0.0001	4392955
50	100,000	.00316	0.00001	1.14×10^{48}

Example

Sampling Normals:

Do Halton Sequence for Λ σ_f, σ_t, ν, ρ;

Summary

- Sampling Methods are slow but effective
- Distributions matter!
- Straight MC is often not the most efficient