



PSUADE Tutorial

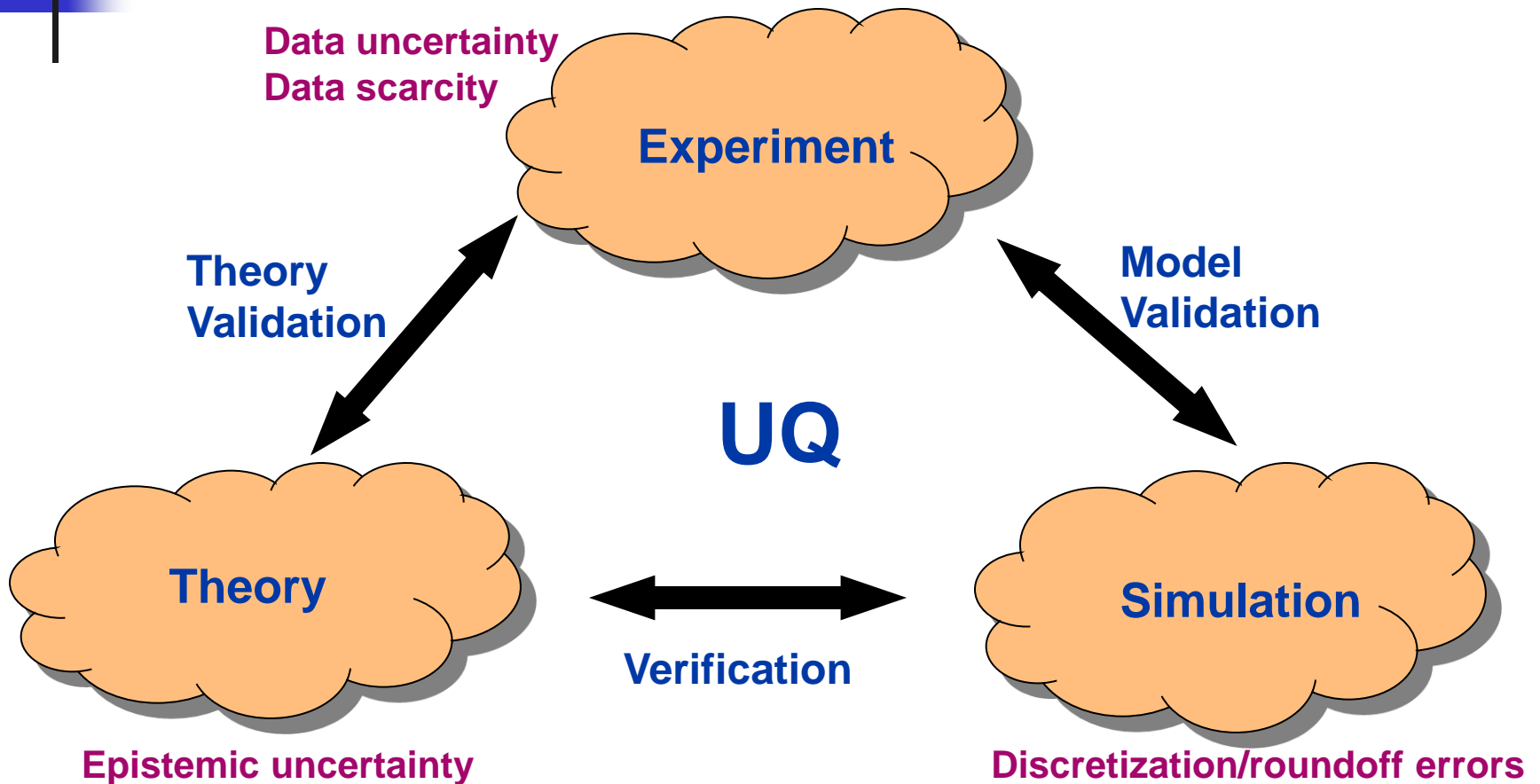
UQ Method Development Team
Lawrence Livermore National Laboratory



Plan

- **Introduction to uncertainty quantification (UQ)**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Global sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**

Robust V&V and UQ are critical in building credible simulation capabilities



Q: How to evaluate whether all 3 agree with each other?

Q: How to get all 3 to agree with each other?

Q: How to evaluate the model's predictive capability?



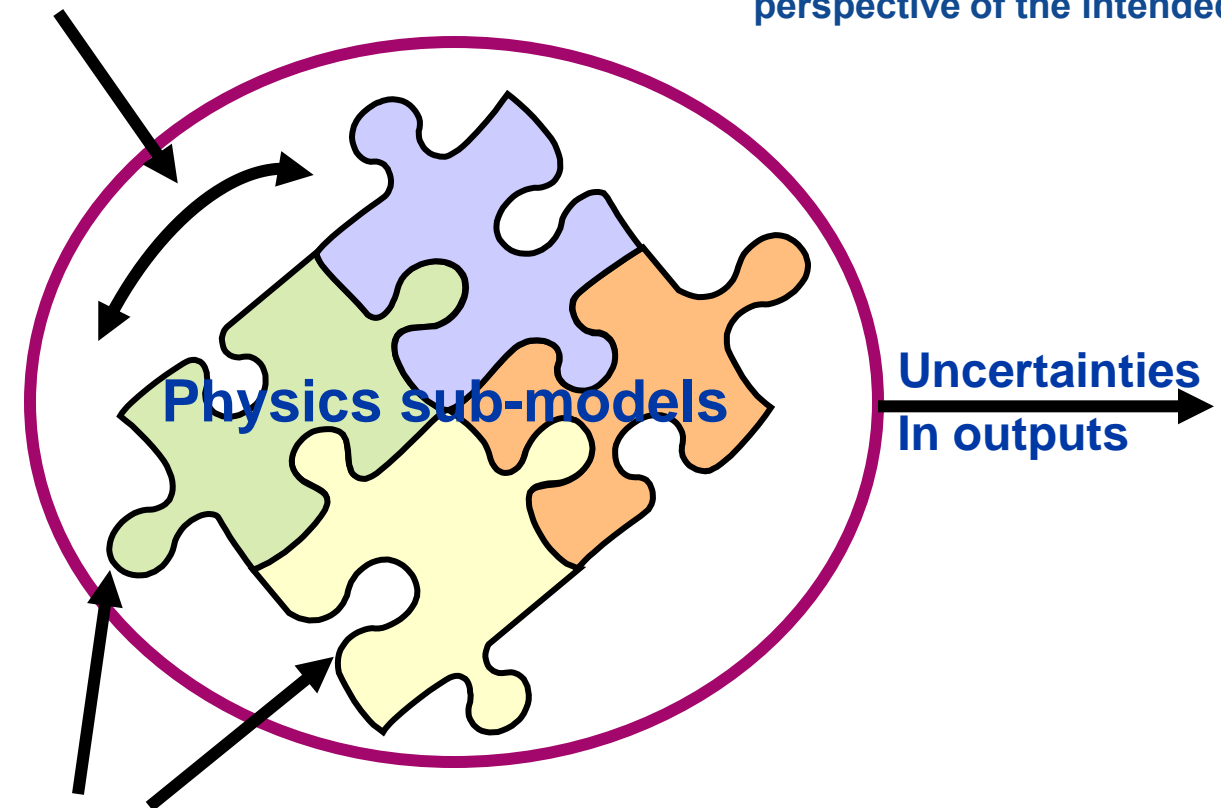
Rigorous verification should precede everything else

- **Computer model = mathematical model + implementation**
- **Verification: ensure the implementation is correct**
 - **Question: Does the solution of a model implementation accurately represent the model solution for its intended use?**
- **Verification activities: (CS and Math, no physics)**
 - **Software quality assurance: regression testing, ...**
 - **Code self-verification (e.g. dynamic consistency checks)**
 - **Grid convergence studies (also time-stepping)**
 - **Solution verification (roundoff error, algorithmic parameters)**
 - **Code to code comparison**
 - **Verification test suite; formal methods**
 - **Bug tracking with Poisson processes**
- **Major question: how to measure coverage?**

UQ is critical in understanding the effects of uncertainties

Validation: "The determination of the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model." – AIAA 1998

Sub-model coupling uncertainties



Sub-model uncertainties

Uncertainties In outputs

Statistics-based model validation
- How about in the event of data scarcity?

Experiment + data with uncertainties





UQ activities that you may be interested in

- Calculate margin and uncertainties (e.g. **QMU**)
- Identify important parameters/set research priorities (**sensitivity analysis**)
- Validate model against experimental data (**validation**)
- Calibrate model parameters to fit data (**calibration**)
- Explore parameter space for important features (**conceptual validation/parameter study**)
- Assess probability of failure in view of parameter uncertainties (**reliability analysis**)
- Inverse UQ, parameter estimation, ...

A typical UQ Analysis

Problem specification (model, variables)

Expert judgment
diligence

Characterize parameter/model uncertainties

Derive credible ranges
Shapes and forms

Parameter Screening: stage I

For nParams $\gg 100$
Single effect analysis

Parameter Screening: stage II

For nParams ~ 100
e.g. use MOAT/GP/MARS
(multi-algorithmic)

Response surface analysis

For expensive models ~ 10
(use MARS, ANN, SVM, GP)

calibration

Quantify uncertainty,
Sensitivity, reliability

Design optimization/
exploration

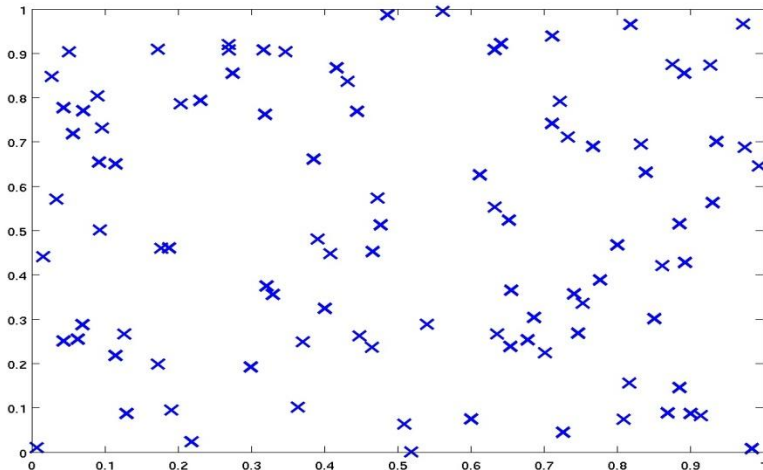


Plan

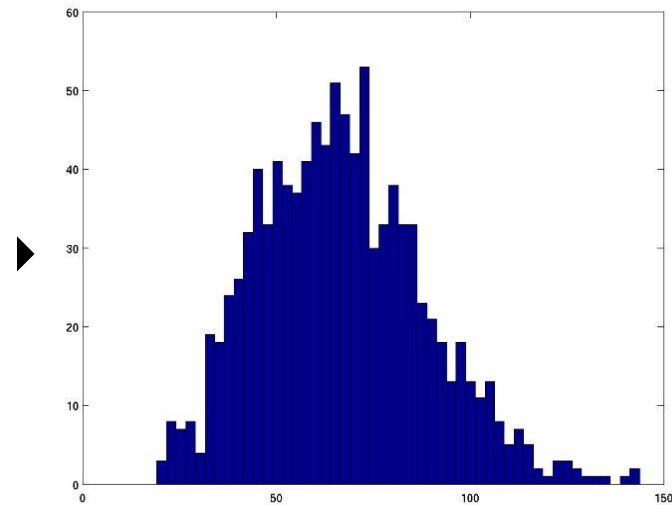
- **Introduction to uncertainty quantification**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Global sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**

A classical approach to calculate uncertainties: Monte Carlo and histogram

1. Create N random sample points in the uncertain parameter space
2. Run the points through the function and gather the Y's
3. Compute basic statistical quantities: mean, std. dev.
4. Bin the Y's and create an output histogram

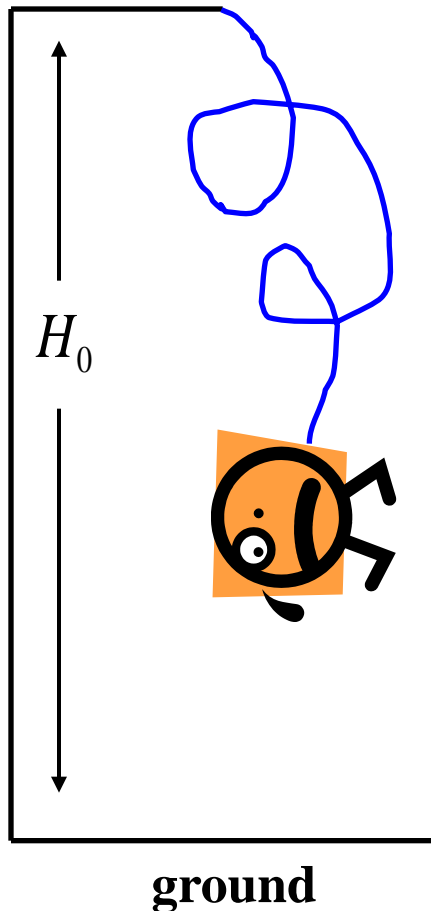


Sample points in parameter space



An example output distribution

Let's use an example to illustrate how to do uncertainty analysis: Bungee Jumping



- To increase excitement, would like the minimum height to be as small as possible but stay alive.
- There are a few uncertainties
 - the height of the platform H_0 (40-60m)
 - the mass of the person M (67-74 kg)
 - the number of strands σ (20-40)
- Objective: In view of uncertainties, assess the risk of safe jumps
- Model: $h = H_0 - (2Mg)/(k\sigma)$
 - k is the elastic constant of the strand
 - g is gravity

Bungee Jumping: we only need to set up 2 PSUADE user files: input file & run file

```
# PSUADE input file (psuade.in)
PSUADE
INPUT
  dimension = 3
  variable 1 H0 = 40 60
  variable 2 M = 67 74
  variable 3 sigma = 20 40
END
OUTPUT
  dimension = 1
  variable 1 H
END
METHOD
  sampling = MC
  num_samples = 1000
END
BEGIN APPLICATION
  driver = ./simulator
END
ANALYSIS
  analyzer method = Moment
END
END
```

```
/* simulator : pseudocode */

read H0, M, sigma from input file

G = 9.8
K = 1.5

H = H0 - (2*M*g)/(K*sigma);

Write H to output file
```



Running PSUADE and analyzing the results are automatic

```
[linux %] psuade psuade.in
```

```
*****
```

```
*** Welcome to PSUADE (version 1.3) ***
```

```
*****
```

```
Psuade: creating interface to user driver
```

```
Psuade: running sample, nSamples = 1000
```

```
.....
```

```
.....
```

```
Psuade: jobs completed = 1000 (out of 1000)
```

```
.....
```

```
* Sample mean = 1.7936e+01
```

```
* Sample std dev = 8.7403e+00
```

M/U ~ 2

```
* Sample skewness = -1.9569e-01
```

```
* Sample kurtosis = 2.5012e+00
```

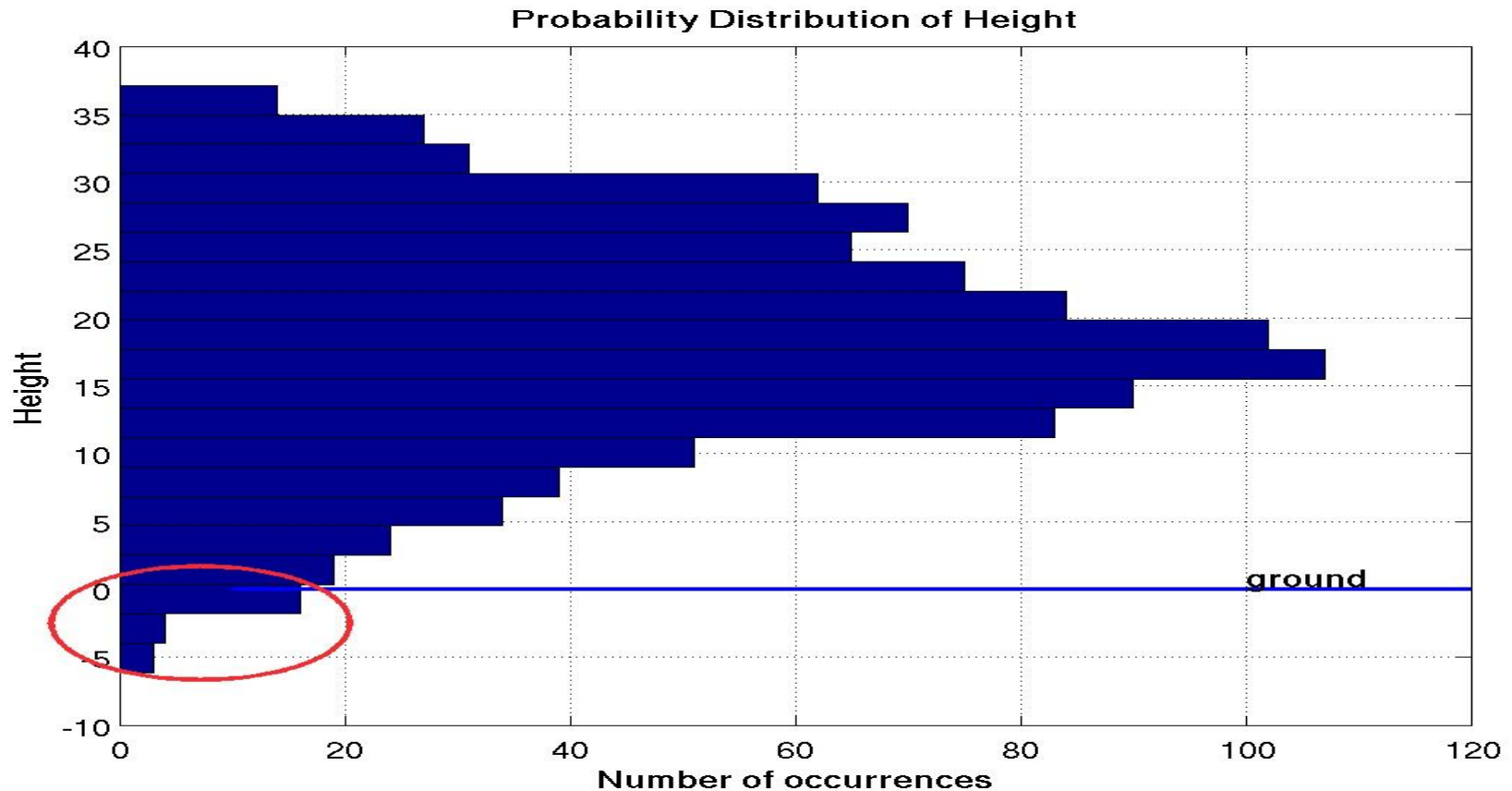
```
-----  
Would you like to plot the output distribution? (y or n) y
```

```
Please enter the matlab file name: distribution.m
```

```
Output distribution plot is now in distribution.m.
```

```
[linux %]
```

Bungee Jumping: probability distribution function shows uncertainty and also risk



Probability of fatal jumps ~ 5% (risk ~ 5%)



Does sample size matter?

[linux %] **psuade psuade.in**

*** Welcome to PSUADE (version 1.3) ***

Psuade: creating interface to user driver

Psuade: running sample, nSamples = 1000 (1000) (MC 2nd time)

.....

.....

Psuade: jobs completed = 1000 (out of 1000)

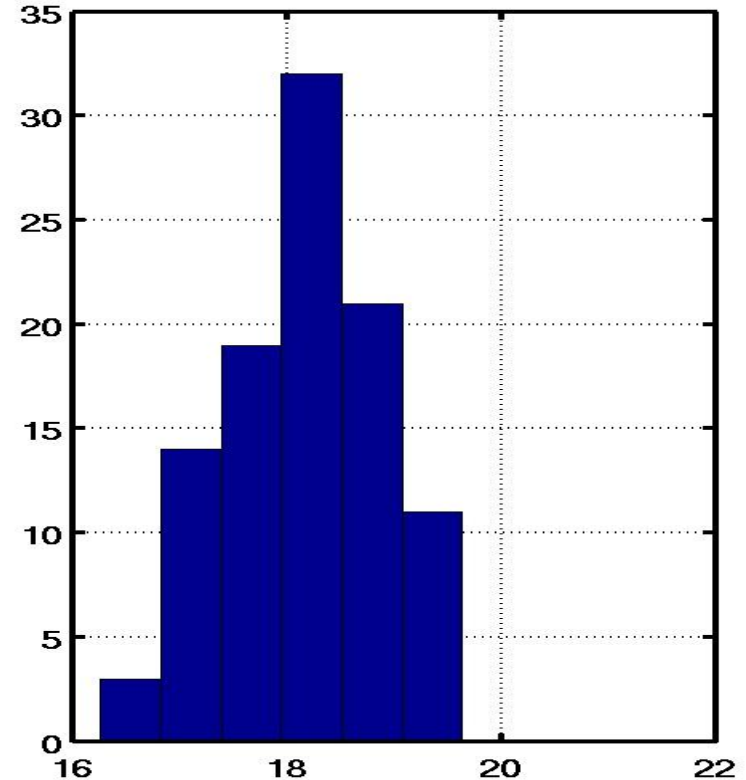
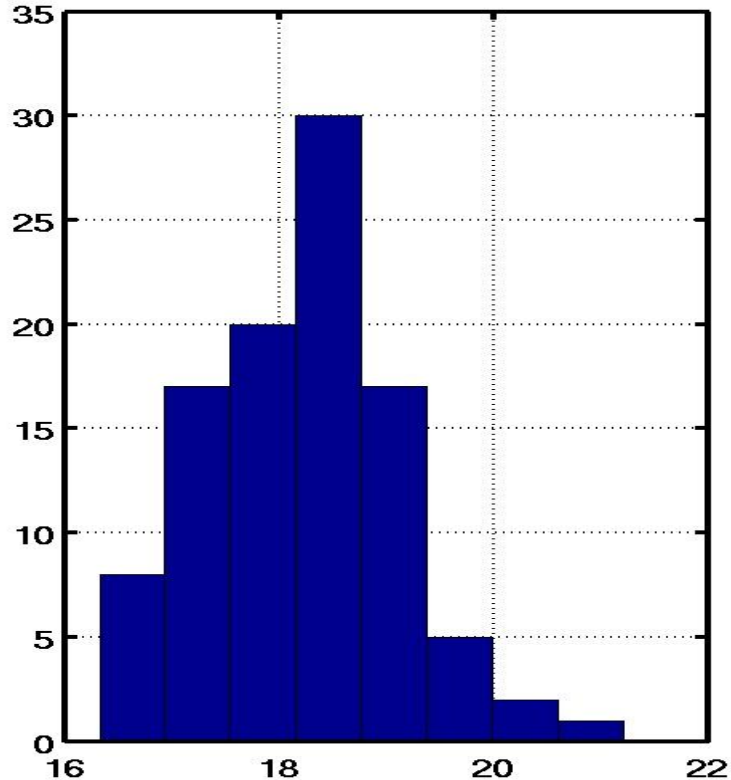
.....

* Sample mean = 1.7936e+01 (1.7776e+01) Slightly different mean
* Sample std dev = 8.7403e+00
* Sample skewness = -1.9569e-01
* Sample kurtosis = 2.5012e+00

Would you like to plot the output distribution? (y or n) **n**

[linux %]

Does the sampling method matter?

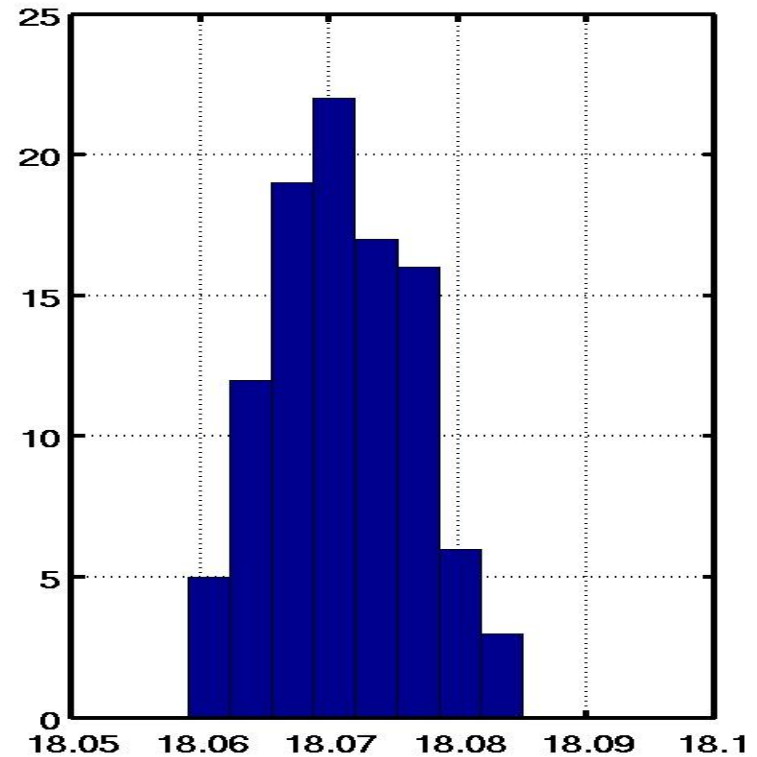
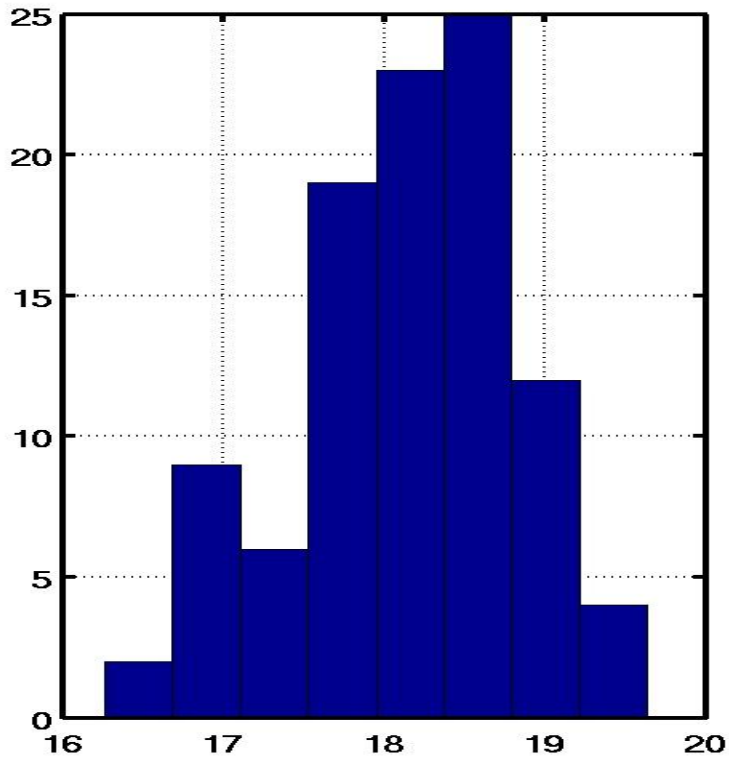


100 Monte Carlo runs (N=100)

100 Monte Carlo runs (N=1000)

Distribution of the sample mean

Does the sampling method matter?

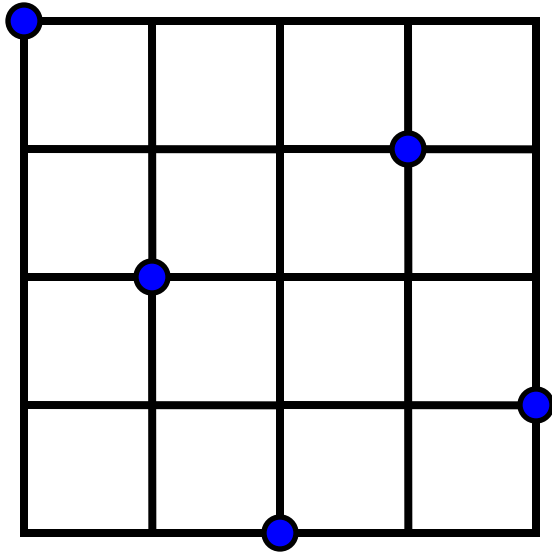


100 Monte Carlo runs

100 Latin hypercube runs

Distribution of the sample mean

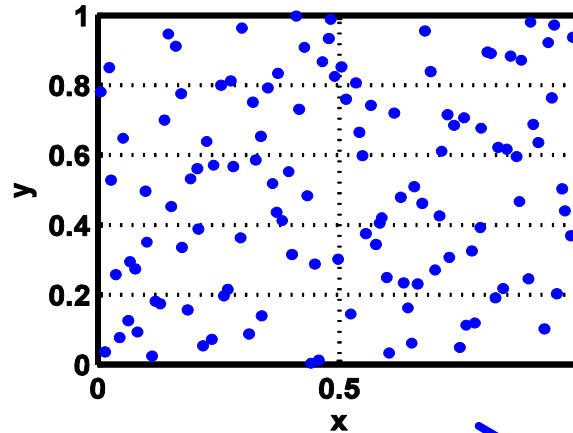
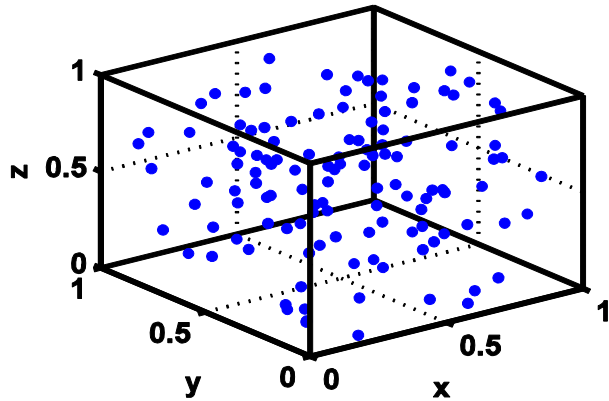
What is Latin Hypercube?



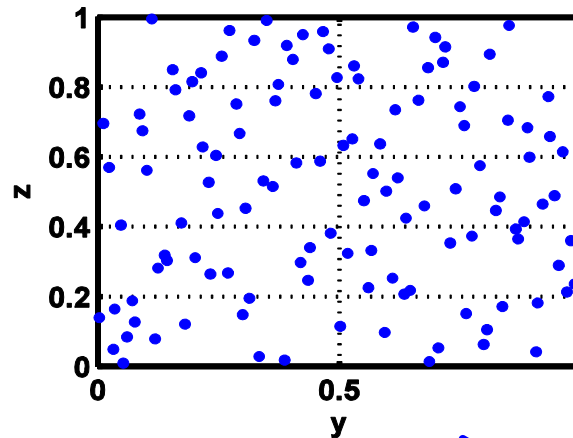
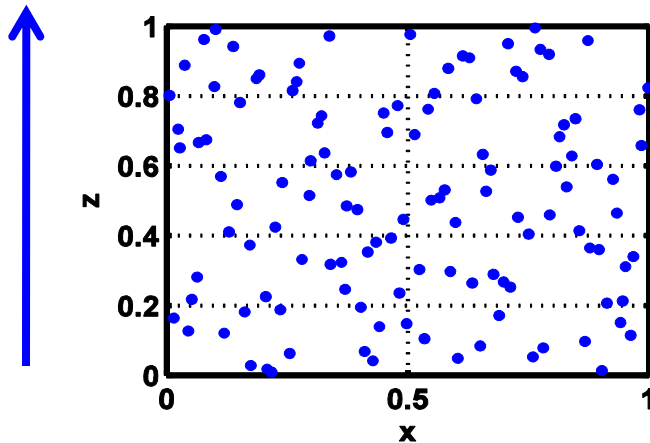
Latin hypercube
(stratified in each dimension)

- space-filling in any one dimension
- faster convergence than MC
 - esp. for monotonic functions
- $LHS(N, m, s) + \text{noise}$
 - **N**: sample size (5 here)
 - **m**: number of parameters
 - **s**: number of symbols
 - **r = N/s**: number of replications
- How to choose sample size?
 - sampling refinement

Stratification properties for Latin hypercube sampling



Stratification in any single dimension

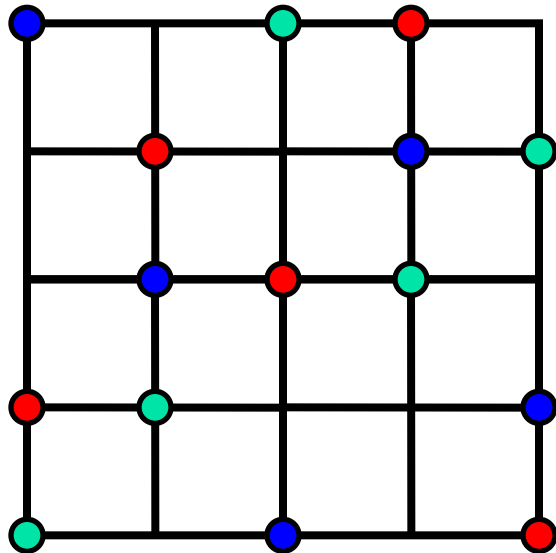


For space-fillingness

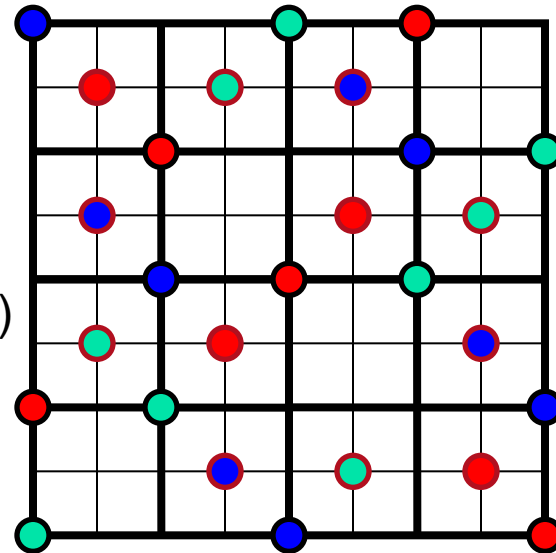
- Maximize min dist
- OA-Latin hypercube
- Centroidal Voronoi tessellation (CVT)

Sampling refinement with Latin Hypercube

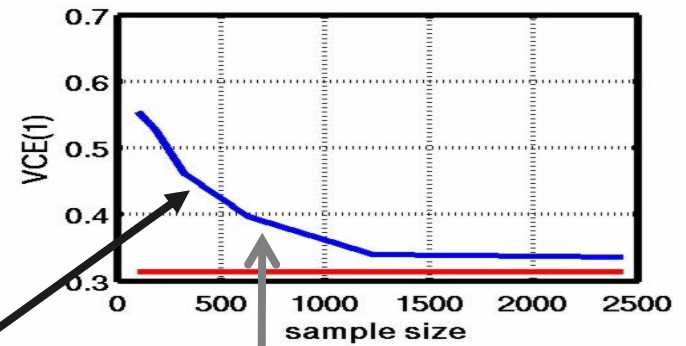
- Objective: to eliminate the need to select an initial sample size for an analysis
- How to use it: e.g. for main effect analysis
 - Refine: $LH(N,m,s) \rightarrow LH(2N-r,m,2s-1)$
 - Iterate and analyze until convergence



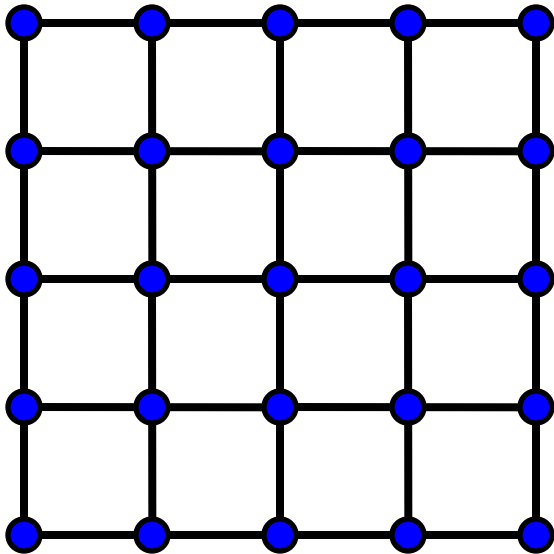
Refinement
→
(2D example)



Numerical results



What is full factorial design?



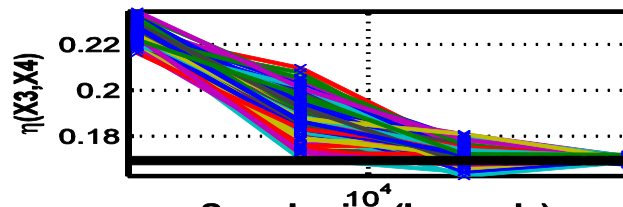
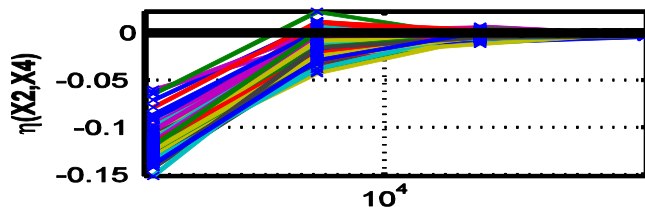
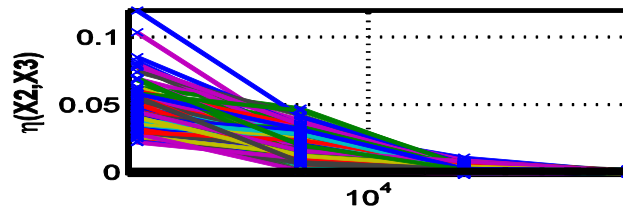
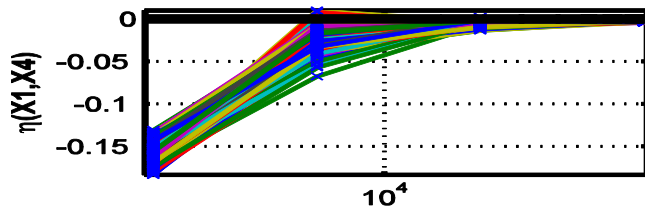
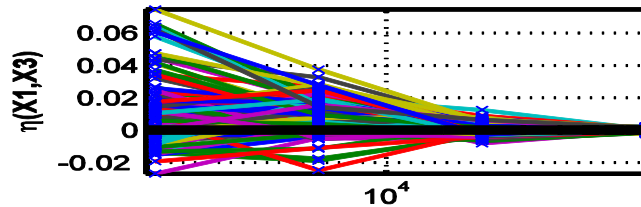
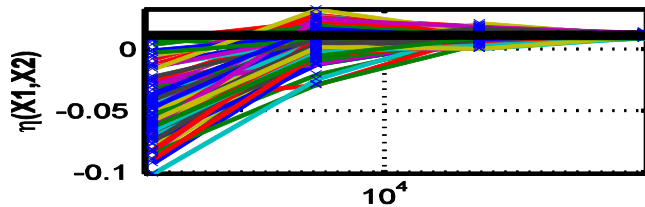
- **space-filling in all dimensions**
- **sample size = s^m**
 - **s: number of levels**
 - **m: number of inputs**
- **can be randomized by small perturbations**
- **can resolve m-way interactions**
- **only suitable for small number of inputs (expensive)**

What is orthogonal array (OA)?

- 4 tuple: OA(N, m, k, s), s=strength
- Use of OA: 2-way conditional variances

$$\sigma(X_k, X_l)^2 = \frac{1}{S^2} \sum_{i=1}^S \sum_{j=1}^S (\bar{Y}_{ij} - \bar{Y})^2$$

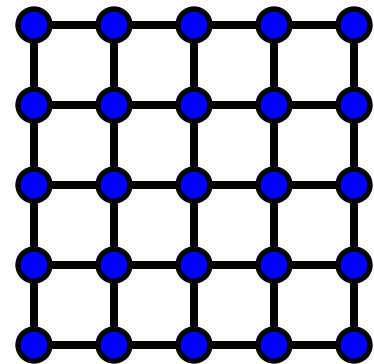
- OA can be refined similarly to LH



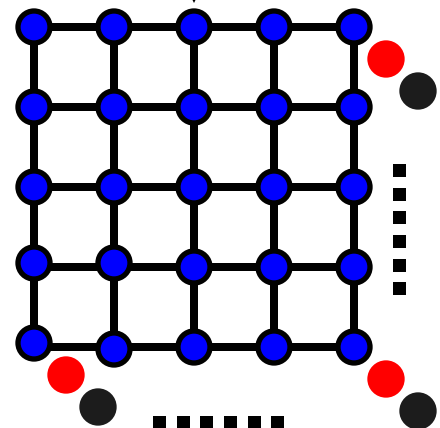
Sample size (log scale)

Sample size (log scale)

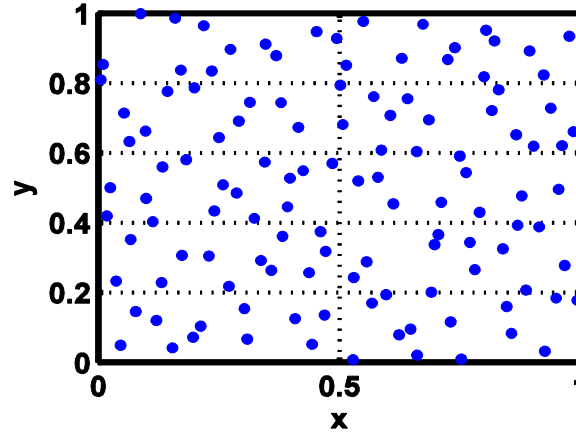
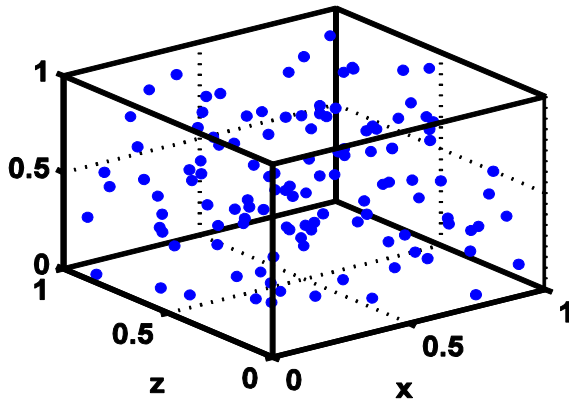
2D projection



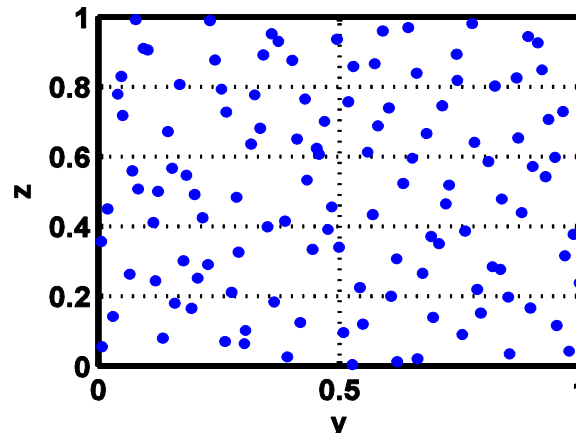
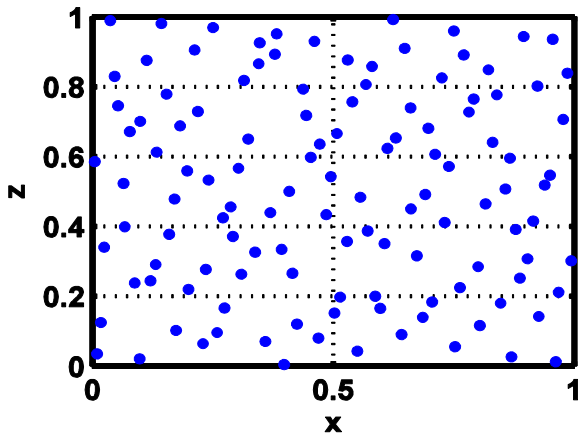
replicate



Multi-dimensional stratification: orthogonal array-based LH



- Stratification in any 1 and 2 dimensions
- OA with strength 3 stratifies in any 3 dim
- Factorial is an extreme case of OA



What is fractional factorial design?

- **Resolution III, IV or V**
- **2 levels → assume linearity for each input**
- **Resolution IV:**
 - for all main effects
 - cannot compute 2-way interactions accurately
- **Resolution V:**
 - for all main effects and two-way interactions
 - thus has more resolving power
 - but larger sample size than Resolution IV
- **An example: 6 inputs, Resolution IV**
 - $X_5 = X_1 X_2 X_3$
 - main effect confounded with 3rd order effects
 - $X_3 X_4 = X_5 X_6$ (2-way confounded)

-1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1
-1	-1	1	-1	1	-1
-1	-1	1	1	1	1
-1	1	-1	-1	1	1
-1	1	-1	1	1	-1
-1	1	1	-1	-1	1
-1	1	1	1	-1	-1
1	-1	-1	-1	1	1
1	-1	-1	1	1	-1
1	-1	1	-1	-1	1
1	-1	1	1	-1	-1
1	1	-1	-1	-1	-1
1	1	-1	1	-1	1
1	1	1	-1	1	-1
1	1	1	1	1	1

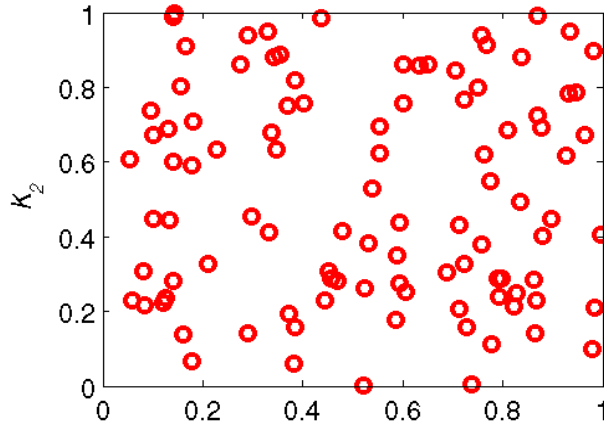


Other sampling designs

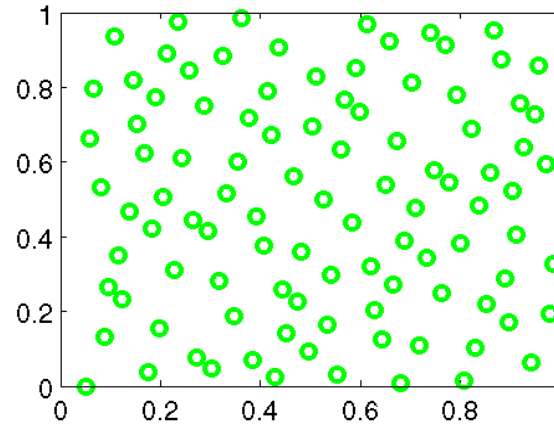
- **Quasi-Monte Carlo: LP-tau, Halton sequence, etc.**
- **Central composite designs (inscribed, circumscribed)**
- **OA-based Latin hypercube (more space-filling than LH)**
- **Plackett-Burman (screening design for linear problems)**
- **Box-Behnken (3 level, fit quadratic)**
- **Morris screening design (screening for nonlinear problems)**
- **Fourier Amplitude Sampling Test (FAST): quantitative SA**
- **Metis (space-filling, less restrictive than full factorial)**

Examples of sampling methods

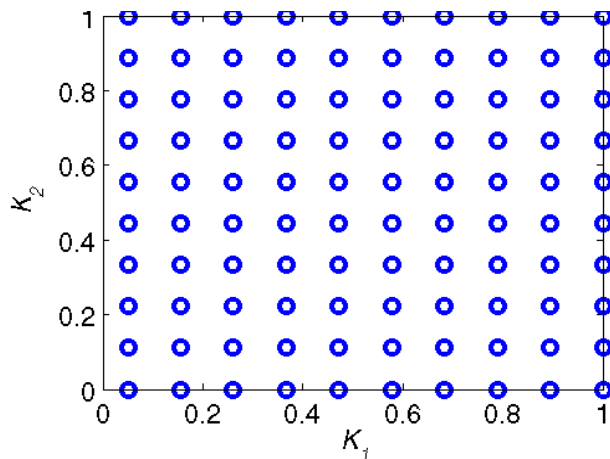
Monte Carlo (MC)



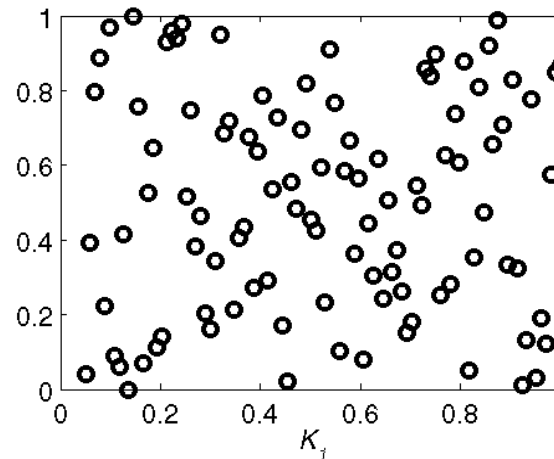
A quasi-random sequence (LPTAU)



Full factorial design (FACT)

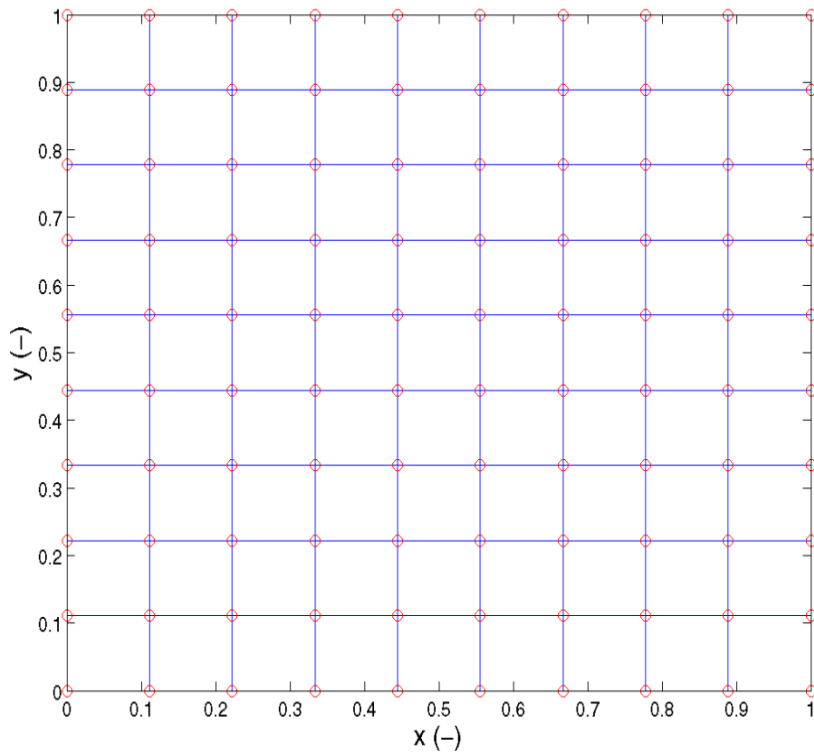


Latin hypercube (LH)



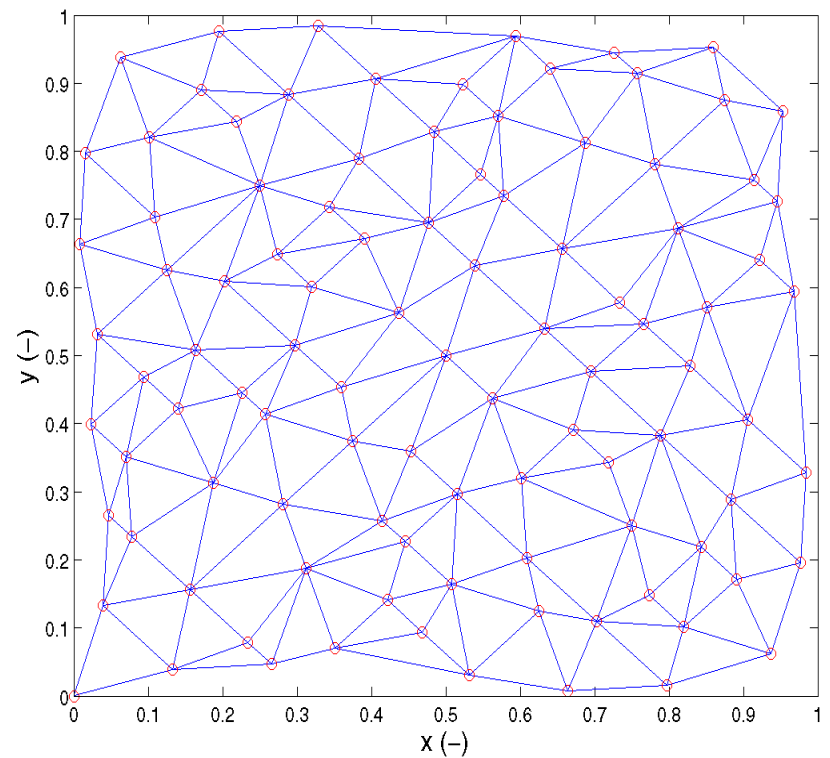
Analogy of sampling methods with grid discretizations

Full factorial design



Finite Difference

A quasi-random sequence



Finite Element



What have we learned so far?

- **Quantifying uncertainties accurately takes many runs**
 - e.g. for MC, converges as $1/\sqrt{N}$
- **Latin hypercube sampling helps, but not sufficient if**
 - the simulation cost is high
 - there are many uncertain parameters
- **Many sampling methods exist for different purposes**
- **How to speed up UQ analysis**
 - dimension reduction (parameter screening)
 - response surface analysis



Plan

- **Introduction to uncertainty quantification**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Global sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**



Variable selection methods are used to down-select the large number of uncertain parameters

Let $X \in \mathbb{R}^m$, design and evaluate $S = \{(X^i, Y^i), i=1, \dots, N\}$
Select $X_G \subset X$ such that $I(X, Y) \cong I(X_G, Y)$ where
 $I(X, Y)$ is the information that X_G brings about Y .

- Objective: identify a small subset of important parameters for further study
 - research prioritization (help us to zoom in on important physics)
 - model reduction (eliminate irrelevant parameters)
- Desirable characteristics of variable selection (screening) methods
 - model independent (can handle nonlinear and non-additive models)
 - allow the exploration of parameters over wide ranges
- These desirable characteristics eliminate linear methods
 - derivative-based methods, linear regression, correlation analysis
 - Fractional factorial and Plackett-Burman designs



Classical Sensitivity Analysis (Pearson/Spearman)

■ Pearson Correlation Coefficient

$$r_{x_j y} = \frac{\sum_{i=1}^{i=N} (x_{ji} - \bar{x}_j)(y_i - \bar{y})}{\left[\sum_{i=1}^{i=N} (x_{ji} - \bar{x}_j)^2 \right]^{1/2} \left[\sum_{i=1}^{i=N} (y_i - \bar{y})^2 \right]^{1/2}}$$

Ratio of co-variance of input (x_j) and output(y) to cross-variances of input (x_j) and output(y)

■ Spearman Correlation Coefficient

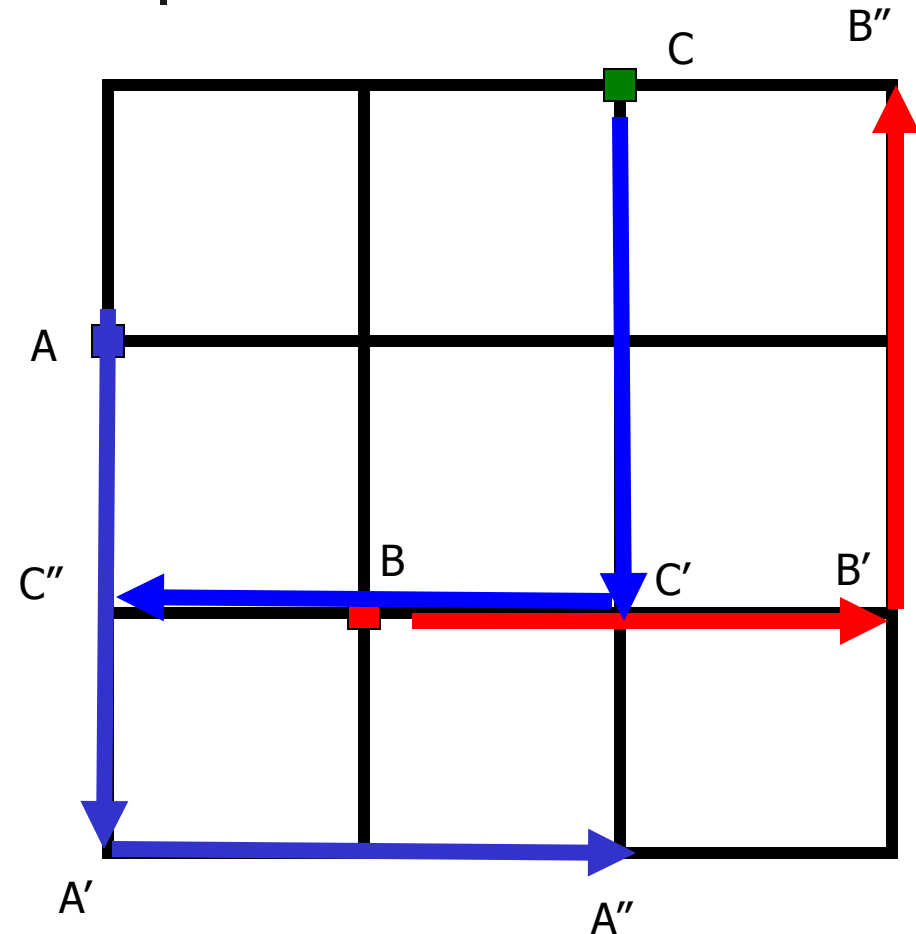
- Replace each entry of Y with its rank and apply Pearson method
- More suitable for nonlinear problems (but monotonic)



Model-independent Variable Selection Methods

- **The Morris screening method**
 - based on sampling the gradients
- **The Delta test**
 - based on nearest-neighbor analysis
- **The Sum-of-trees method**
 - belongs to the class of tree-based methods
- **MARS-based importance analysis**
 - based on analysis from spline interpolation

The Morris screening method



1. Start at a random point (A)
 2. Create the next point by perturbing one input (A')
 3. Create the next point by perturbing another input (A'')
- Repeat step 1-3 r times (B,C..)
 - Form r gradients for each input and compute modified means and standard deviations
 - Plot mean vs standard dev. for each input \rightarrow screening diagram



How does the Morris screening method work?

Gradient of response w.r.t the j-th input

$$z_j = \frac{y(x_1, x_2, \dots, x_j + \Delta x_j, \dots, x_m) - y(x_1, x_2, \dots, x_j, \dots, x_m)}{\Delta x_j}$$

Vector of gradients: with m input parameters

$$\mathbf{Z}_r = (z_1, z_2, \dots, z_m)$$

Collection of gradient vectors (R paths or replications):

$$\Omega = \{ \mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_R \} \quad \bar{z}_j = \frac{1}{R} \sum_{i=1}^R |z_{ij}|$$

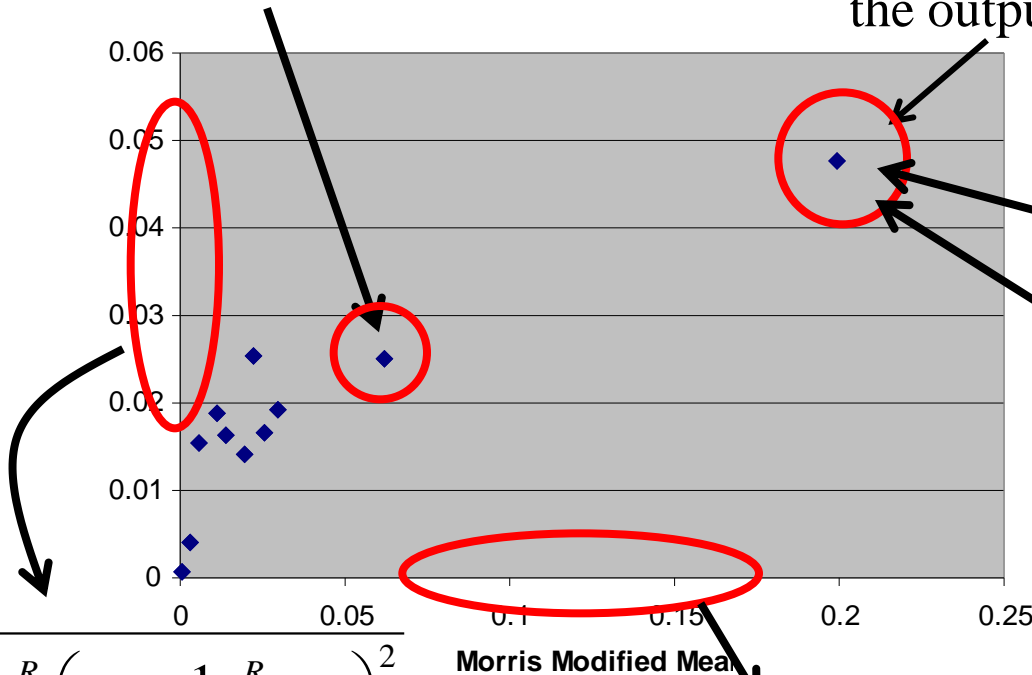
Study the statistics (mean and standard deviation) of Ω

Interpretation: Screening diagram is a distillation of the Morris screening data

Each point refers to one particular input parameter

Each point represents the average “effect” of that particular input on the outputs

Decision on how to cut should be decided by scientists



based on R points
($R = \#$ replicates)

$$\sigma_j = \sqrt{\frac{1}{R-1} \sum_{i=1}^R \left(Z_{ji} - \frac{1}{R} \sum_{i=1}^R Z_{ji} \right)^2}$$

Morris Modified Mean

$$\bar{Z}_j = \frac{1}{R} \sum_{i=1}^R |Z_{ji}|$$

Note: mean is based on absolute value of the output

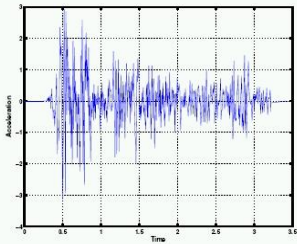
Large σ = non-linear relationship or inter-parameter interactions

Large mean = “sensitive” parameter

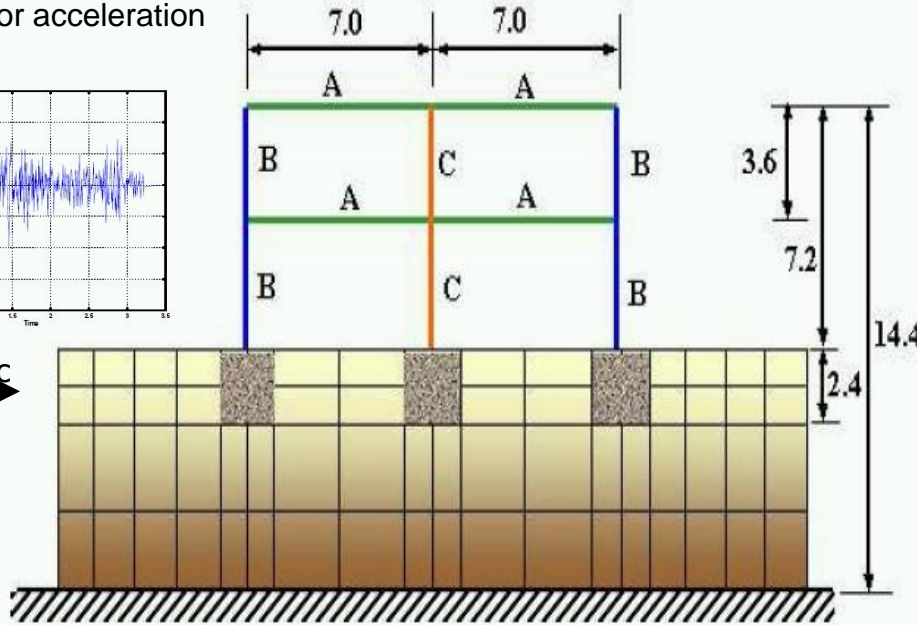
An Example: a soil-foundation-structure-interaction model

- Response: Interstory drift
- max roof drift/bldg height
 - max story drift/story height
 - max floor acceleration

(UCSD model)
OpenSees/1940 El Centro



Seismic wave →



Material parameters and uncertainties

	Parameter	Distribution	Mean	range
Upper Cover Concrete	f_c	Uniform	27588.5	± 0.2 * Mean
	e_{c0}	Uniform	0.002	± 0.2 * Mean
	e_{cu}	Uniform	0.008	± 0.2 * Mean
Upper Core Concrete	f_c	Uniform	34485.6	± 0.2 * Mean
	f_{cu}	Uniform	20691.4	± 0.2 * Mean
	e_{c0}	Uniform	0.004	± 0.2 * Mean
	e_{cu}	Uniform	0.014	± 0.2 * Mean
Upper Steel	E	Uniform	$2.0e8$	± 0.033 * Mean
	S_y	Uniform	248200	± 0.106 * Mean
	Hkin	Uniform	1.613	± 0.2 * Mean
Foundation	E	Uniform	$2.0e7$	± 0.2 * Mean
Soil Layer #1	G	Uniform	54450	± 0.3 * Mean
	τ_{max}	Uniform	33.0	± 0.25 * Mean
Soil Layer #2	G	Uniform	33800	± 0.3 * Mean
	τ_{max}	Uniform	26.0	± 0.25 * Mean
Soil Layer #3	G	Uniform	61250	± 0.3 * Mean
	τ_{max}	Uniform	35.0	± 0.25 * Mean
Soil Layer #4	G	Uniform	96800	± 0.3 * Mean
	τ_{max}	Uniform	44.0	± 0.25 * Mean

Questions related to uncertainty quantification (UQ):

1. Characterize inter-story drift uncertainties due to parameter uncertainties
2. Rank the parameters in importance (sensitivity analysis SA)



Running Morris Analysis using PSUADE

```
[linux %] psuade
```

```
psuade> Load psData
```

```
psuade> moat
```

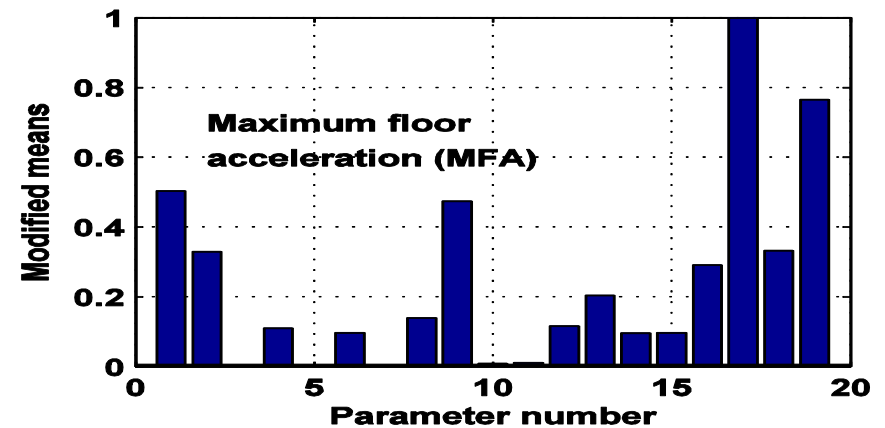
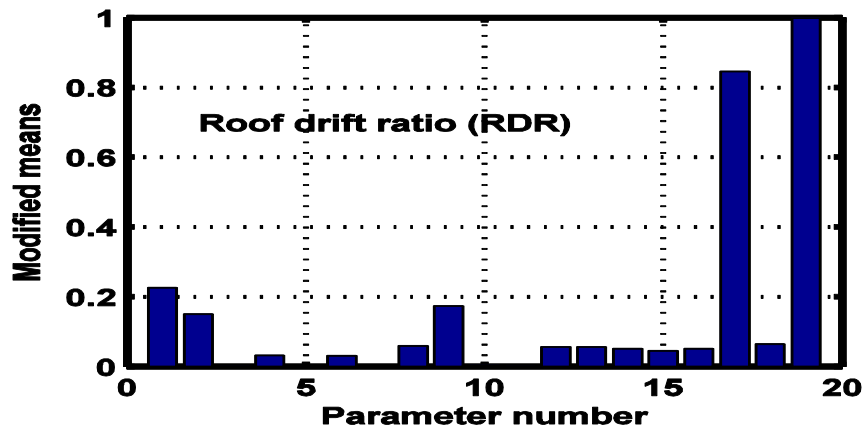
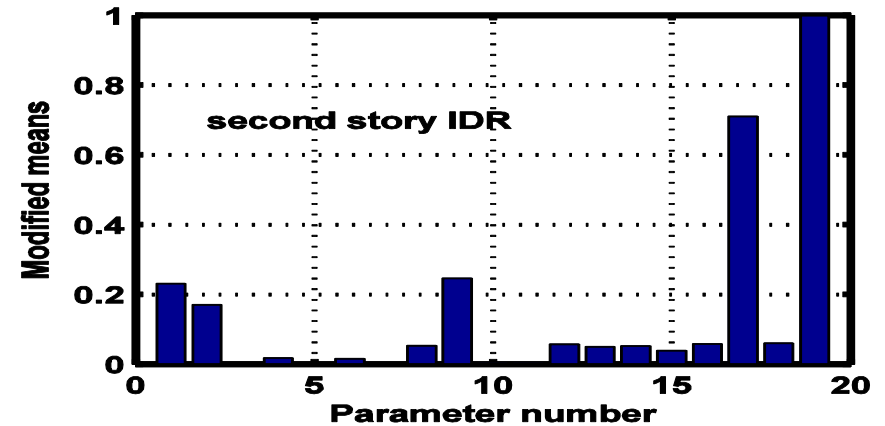
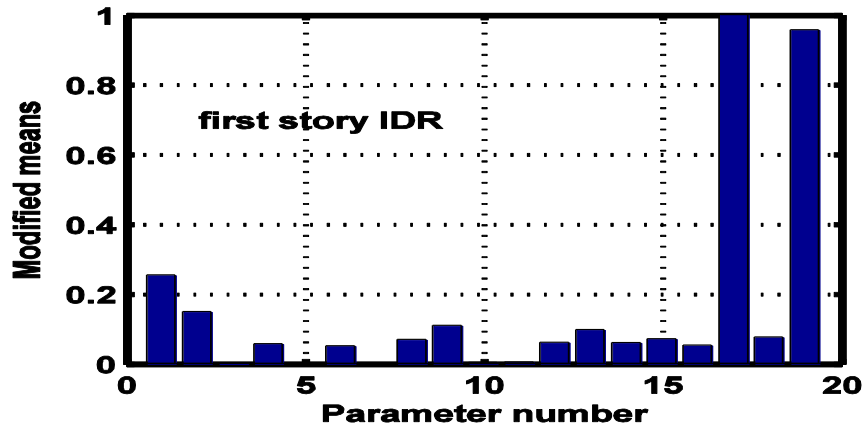
```
moat: Morris screening analysis
```

```
***** MOAT Analysis *****
```

```
Input 1 (mod. mean & std) = 1.2781e-02 2.9352e-03  
Input 2 (mod. mean & std) = 7.1482e-03 1.7959e-03  
Input 3 (mod. mean & std) = 7.5600e-06 1.5736e-05  
Input 4 (mod. mean & std) = 2.1727e-03 1.0429e-03  
Input 5 (mod. mean & std) = 0.0000e+00 0.0000e+00  
Input 6 (mod. mean & std) = 1.8366e-03 7.6315e-04  
Input 7 (mod. mean & std) = 0.0000e+00 0.0000e+00  
Input 8 (mod. mean & std) = 3.6514e-03 1.3754e-03  
Input 9 (mod. mean & std) = 1.8199e-02 7.2860e-03  
Input 10 (mod. mean & std) = 4.0640e-04 2.6256e-04  
Input 11 (mod. mean & std) = 2.4656e-04 2.5489e-04  
Input 12 (mod. mean & std) = 3.8643e-03 2.0851e-03  
Input 13 (mod. mean & std) = 5.0526e-03 4.8351e-03  
Input 14 (mod. mean & std) = 4.3793e-03 1.7473e-03  
Input 15 (mod. mean & std) = 3.6571e-03 3.1795e-03  
Input 16 (mod. mean & std) = 1.7720e-02 8.4490e-03  
Input 17 (mod. mean & std) = 1.6789e-02 1.7281e-02  
Input 18 (mod. mean & std) = 2.1844e-02 1.0673e-02  
Input 19 (mod. mean & std) = 2.6873e-02 2.3267e-02
```

```
*****
```

Morris screening results for the SFSI2D problem



- 1: CoverC
- 2: CoverEC0
- 9: Steel σ
- 17: Soil T3
- 19: Soil T4

The Gamma Test and Delta Test

- Assumption: if two points are close together in input space then their outputs should be close in the output space. If the outputs are not close together, we consider this difference is because of noise.
- Assumption: the underlying input-output relationship is of the form $Y=F(X)=F(X_1, \dots, X_m)+\varepsilon$ where ε has zero mean and bounded variance.
- Let the set of data points be $\{(X^i, Y^i), 1 \leq i \leq N\}$
- Let $X_{N(i,k)}$ denote the k-th nearest neighbor of X^i
- Let

$$\delta(k) = \frac{1}{N} \sum_{i=1}^N (X_{N(i,k)} - X^i)^T (X_{N(i,k)} - X^i), \gamma(k) = \frac{1}{2N} \sum_{i=1}^N |Y_{N(i,k)} - Y_i|^2$$
- Γ statistics can be calculated as y-intercept of the regression line for $\{\delta_M(k), \gamma_M(k), k=1, \dots, p\}$
- The theory says that $\gamma_M(k) \rightarrow \text{Var}(\varepsilon)$ as $\delta_M(k) \rightarrow 0$



An Enhanced Delta Test for Variable Subset Selection

▪ Again, let $Y = F(X) = F(X_1, \dots, X_m) + \varepsilon$

▪ Let the set of data points be $\{(X^i, Y^i), 1 \leq i \leq N\}$

▪ Let the Delta metric be
$$\delta(k) = \frac{1}{N} \sum_{i=1}^N (X_{N(i,k)} - X^i)^T (X_{N(i,k)} - X^i)$$

▪ The original Delta test is: the best variable subset S^* is such that

$$S^* = \arg \min_{S \subset X} \delta_S(1)$$

where $\delta_S(1)$ is the Delta metric restricted to the variable subset space S

▪ The improved Delta test consists of

- using additional neighbors ($k=1,2,3$)
- choosing instead the best 50 subsets and using them for scoring
- assessing the final choice using forward sweep

Running Delta Test using PSUADE

```
[linux%] psuade
```

```
psuade> Load psData
```

```
psuade> delta_test
```

```
=====
```

Current best solution for output 1:

```
0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 = 1.273683e-04
```

```
0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 1 1 = 1.584548e-04 (1 of 100)
```

```
.....
```

```
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 = 6.067841e-05 (100 of 100)
```

```
*****
```

Final Selections (based on 3 neighbors) =

```
Rank 1 => 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 : delta = 6.0678e-05
```

```
.....
```

```
-----
```

Order of importance (based on 20 best configurations):

```
(D)Rank 1 : input 19 (score = 100 )
```

```
(D)Rank 2 : input 9 (score = 100 )
```

```
(D)Rank 3 : input 18 (score = 100 )
```

```
(D)Rank 4 : input 17 (score = 100 )
```

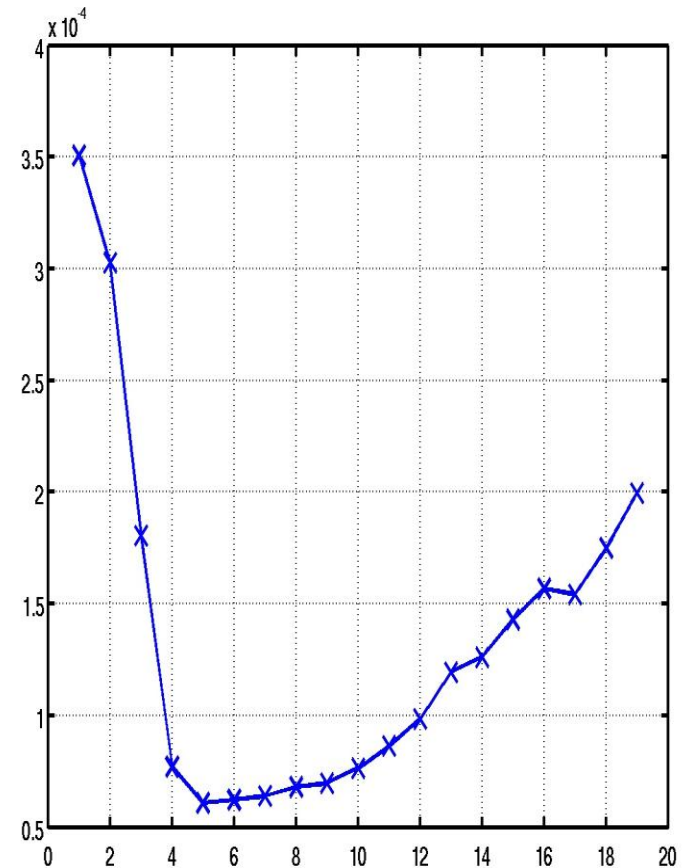
```
(D)Rank 5 : input 16 (score = 100 )
```

```
(D)Rank 6 : input 1 (score = 49 )
```

```
(D)Rank 7 : input 10 (score = 44 )
```

```
(D)Rank 8 : input 11 (score = 24 )
```

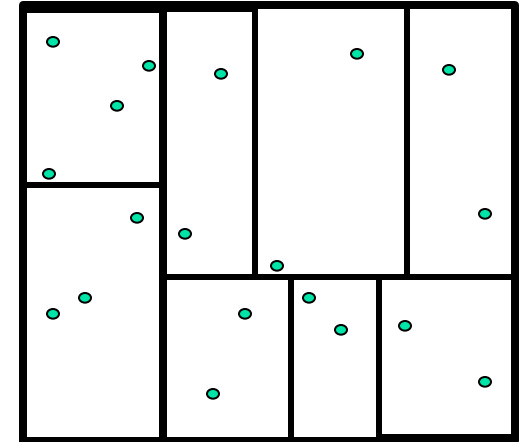
```
.....
```



Order: 19, 9, 18, 17, 16, 1, 10, 11, 15, 2

Tree-based Methods

- Based on creating a binary tree (unbalanced)
- Criteria for splitting: use impurity function
 - residual sum of squares
 - ratios of means and variances of sub-trees
- Splitting criterion: maximum decrease in impurity
- Stopping criteria:
 - minimum number of data points per terminal nodes
 - residual sum of squares falls below a threshold
- Sum-of-trees
 - use bootstrapped samples and average (*)
 - use boosting and average
- Ranking criterion:
 - total number of splittings (with scaling at each level) for each input
 - use variance of the number of splittings as error bar





Running Delta Test using PSUADE

```
[linux %] psuade
```

```
psuade> Load psData
```

```
psuade> sot_sa
```

```
*****
```

```
* SumOfTrees screening rankings (with bootstrapping)
```

```
*****
```

```
* Rank 1 : Input = 19 (score = 100.0)
```

```
* Rank 2 : Input = 17 (score = 92.6)
```

```
* Rank 3 : Input = 9 (score = 61.6)
```

```
* Rank 4 : Input = 18 (score = 61.1)
```

```
* Rank 5 : Input = 16 (score = 39.8)
```

```
* Rank 6 : Input = 1 (score = 11.9)
```

```
* Rank 7 : Input = 10 (score = 9.6)
```

```
* Rank 8 : Input = 2 (score = 7.0)
```

```
* Rank 9 : Input = 13 (score = 5.6)
```

```
* Rank 10 : Input = 5 (score = 4.6)
```

```
* Rank 11 : Input = 3 (score = 3.7)
```

```
* Rank 12 : Input = 4 (score = 3.5)
```

```
* Rank 13 : Input = 8 (score = 3.5)
```

```
* Rank 14 : Input = 14 (score = 3.3)
```

```
* Rank 15 : Input = 11 (score = 3.1)
```

```
.....
```

```
*****
```

```
psuade>
```



Numerical Results (2D SFSI Problem)

Method	Top 7 parameters
SPEA	17,19,18,9,16,2,14
Morris	19,18,9,16,17,1,2
MARS	19,17,18,9,16,1,2
Delta Test	9,17,19,16,18,1,13
SumOfTrees	19,17,18,9,16,1,2

Sample size used: MC with N=600

Variable 13: ranked by other methods as (8,8,8,9,13)

Variable 14: ranked by other method as (9,11,10,12,18)



Plan

- **Introduction to uncertainty quantification**
- **Uncertainty analysis and screening**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Global sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**



Response Surfaces

- response surfaces are representations of the model output everywhere the parameter space

$$Y = F(\mathbf{X}) \approx \hat{F}(\mathbf{X}) \text{ in } \Omega$$

- Other names
 - surrogate model
 - (stochastic) emulator
- Basic ingredients of a response surface analysis
 - a sample (input-output pairs, space-filling)
 - a response surface fitting method



Response surfaces (surrogates, emulators,) are very useful for the UQ of multi-physics models

- **Multi-physics models are generally expensive to evaluate (many CPU hours)**
- **Robust uncertainty quantification (forward/inverse uncertainty assessment, sensitivity analysis) needs many sample points**
- **Idea: use sampling and assumptions about the function f to construct an approximate mapping**
- **Challenges**
 - **Parameter space large (>10)**
 - **Near-singularities/discontinuities/noise**

Definition:

Evaluate $S = \{ (X^i, Y^i), i=1, \dots, N, X_i \in \mathbb{R}^m, Y^i \in \mathbb{R} \}$
Find $f \in F$ (hypothesis function space) such that
 $V(S, f)$ (some error measure) is minimized.



Response Surface Fitting Method

- Parametric:
 - linear regression, quadratic, cubic, quartic, etc.
 - special polynomial: Legendre
 - nonlinear regression functions
- Nonparametric:
 - multivariate adaptive splines (MARS) + bootstrap aggr
 - artificial neural network
 - Gaussian process (GP, kriging, treed-GP)
 - Support vector machines
 - Sum-of-trees
 - Many others: e.g. wavelet, ...
- Selection depends on knowledge of the function and sample size (e.g. GP is very expensive)



How to create response surfaces?

1. Choose a sampling method (LP-tau, Metis, LH, etc.)
2. Run the simulator with the sample
3. Use response surface check to see goodness of fit
 - examine training errors
 - examine cross validation errors
4. If errors are not acceptable, add more points
5. Create a FF IV design to sample some corners
 - to test the robustness against extrapolation
6. Use 'rstest' to examine extrapolation errors
7. If good, add FF design and create new response surface

How to create response surfaces: an example

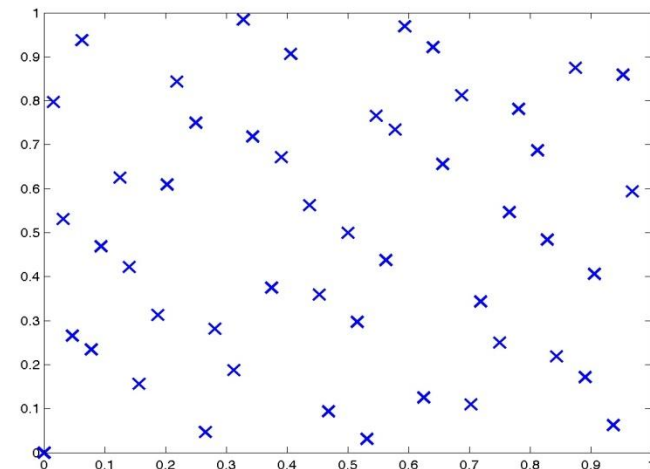
```
# PSUADE input file (psuade.in)
PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0 1
  variable 2 X2 = 0 1
END
OUTPUT
  dimension = 1
  variable 1 Y
END
METHOD
  sampling = LPTAU
  num_samples = 50
END
BEGIN APPLICATION
  driver = ./simulator
END
ANALYSIS
END
END
```

```
/* simulator : pseudocode */
```

```
read X1, X2 from input file
```

$$Y = X1 + X2 * X2 \quad Y = x_1 + x_2^2$$

```
Write Y to output file
```



Generating response surface using PSUADE

```
[linux %] psuade psuade.in
```

```
....
```

```
[linux %] mv psuadeData psData
```

```
[linux %] psuade
```

```
psuade> load psData
```

```
psuade> rs2
```

```
Grid resolution ? (32 – 256) 128
```

```
Available response surface tools:
```

```
0. MARS
```

```
1. Linear regression
```

```
2. Quadratic regression
```

```
3. Cubic regression
```

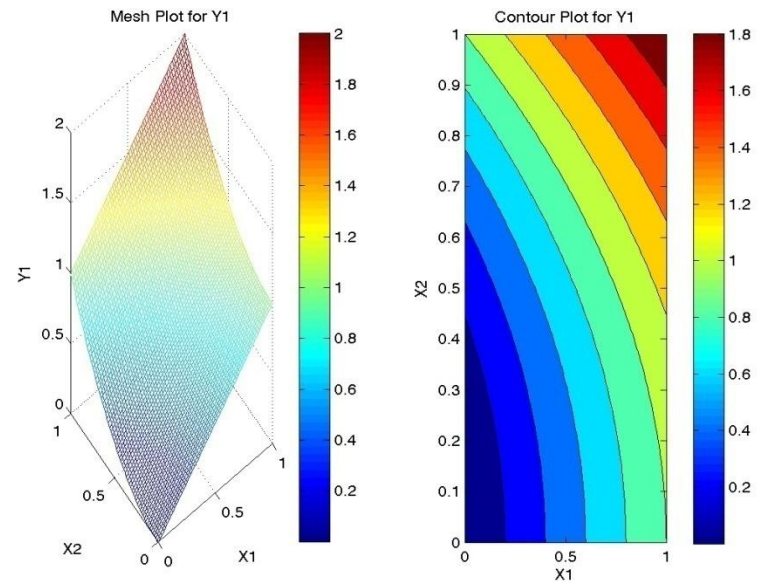
```
.....
```

```
Please enter your choice ? 0
```

```
matlabrs2.m is now available for response surface and contour plots
```

```
psuade> quit
```

```
[linux %]
```



Check response surface quality

```
[linux %] psuade
psuade> load psData
psuade> rscheck
```

Available response surface tools:

0. MARS

1. Linear regression

.....

Please enter your choice ? 0

.....

RSFA: L 0: interpolation error on training set =

L1n error = 4.869e-03 (unscaled), 8.005e-02(scaled)

Avg error = -1.412e-08 (unscaled), 2.046e-02(scaled)

RMS error = 6.138e-03 (unscaled), 6.580e-01(scaled)

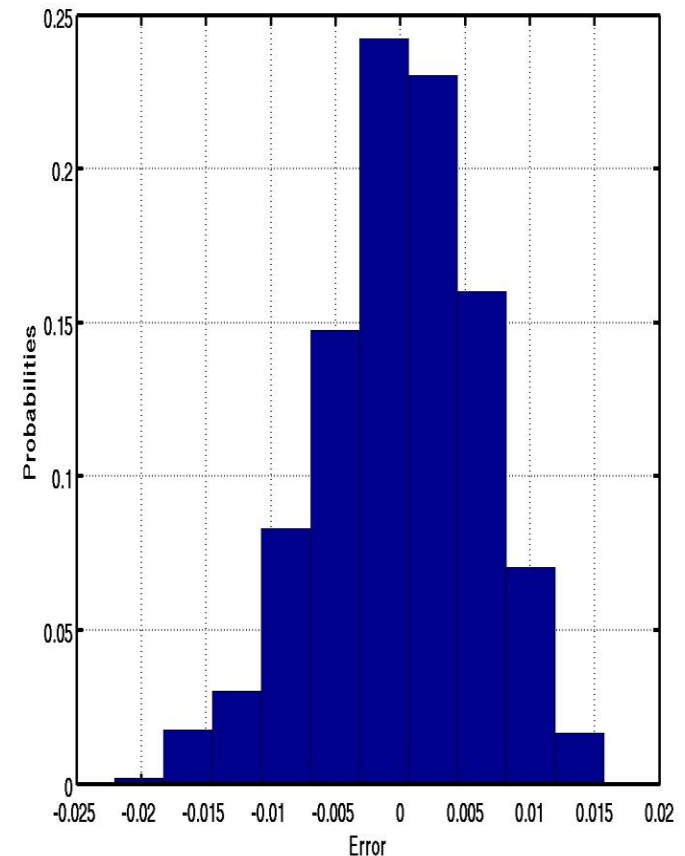
Max error = 2.207e-02 (unscaled, BASE=2.469e-02)

Max error = 1.900e+01 (scaled, BASE=1.428e-04)

Based on 1024 training points (total=1024).

Distribution of interpolation error is in psuade_rsfa_err.m

Perform cross validation ? (y or n) y



Check response surface quality (cont)

Enter the number of groups to validate : (2 - 1024) **10**

RSFA: number of CV groups = 10

RSFA: L 1:cross validation (CV) begins...

Random selection of leave-out groups ? (y or n) **y**

RSFA:: L 1:CV processes 1 out of 10.

.....

RSFA:: L 1:CV processes 2 out of 10.

.....

RSFA: final CV error = $5.025e-03$ (L1n unscaled)

RSFA: final CV error = $2.142e-01$ (L1n scaled)

RSFA: final CV error = $2.037e-01$ (RMS unscaled)

RSFA: final CV error = $7.498e+01$ (RMS scaled)

RSFA: final CV error = $2.326e-02$ (Max unscaled, BASE= $1.448e+00$)

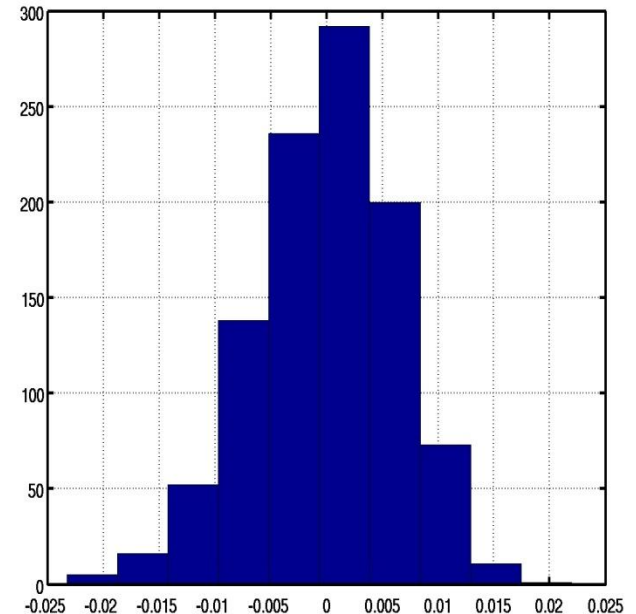
RSFA: final CV error = $4.952e+01$ (Max scaled, BASE= $1.428e-04$)

RSFA: L 1:cross validation (CV) completed.

CV error file is RSFA_CV_err.m (CV error for each point).

AnalysisManager: analysis error = $4.24e-04$ <? $1.00e+00$

psuade> **quit**





Regression Analysis

- **Linear regression:**
$$\hat{Y}_i = \sum_{k=1}^n b_k x_{ki} + b_0 + \varepsilon_i \Rightarrow \hat{Y} = XB + E$$

- **Weighted approach gives**
$$B = (X'WX)^{-1} X'WY$$

- **Compute sum of squares statistics**

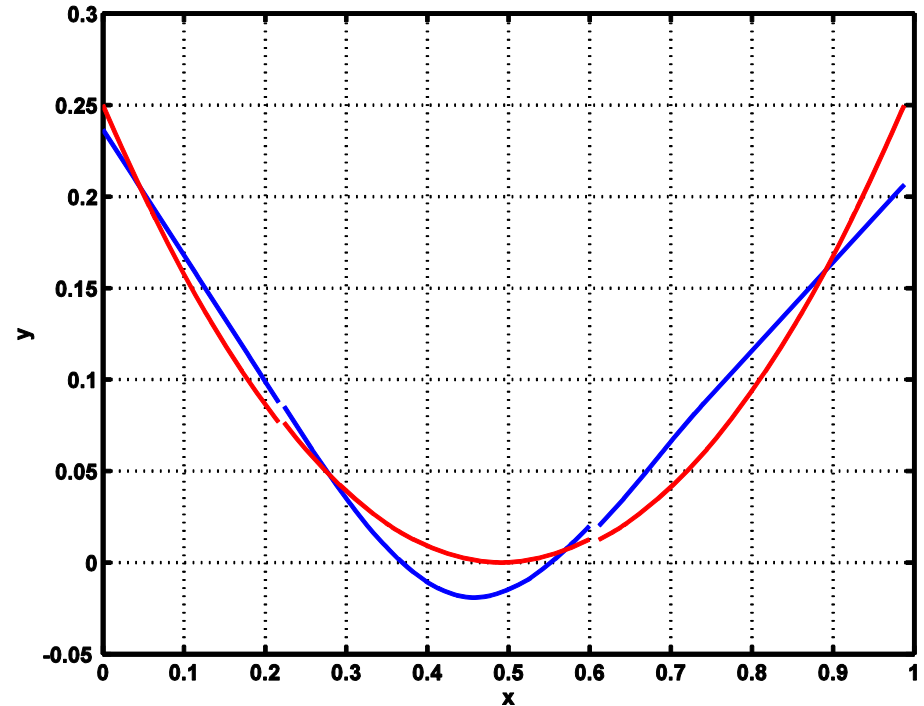
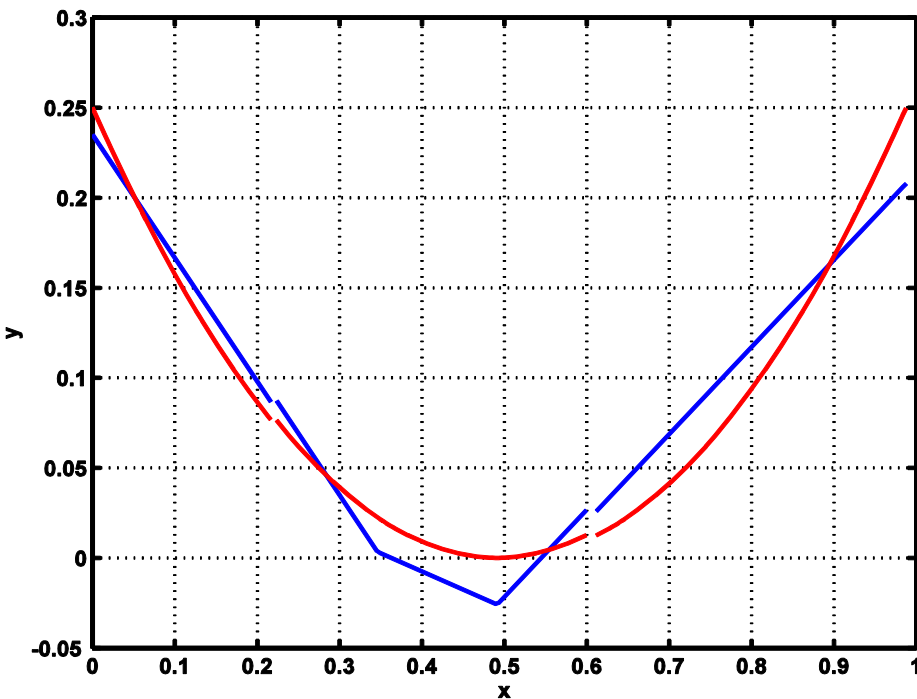
$$SS_{tot} = \sum_{i=1}^N (Y_i - \bar{Y})^2 \quad SS_{reg} = \sum_{i=1}^N (\hat{Y}_i - \bar{Y})^2$$

- **To quantify how close the regression model matches the data, we use**

$$R^2 = \frac{SS_{reg}}{SS_{tot}}$$

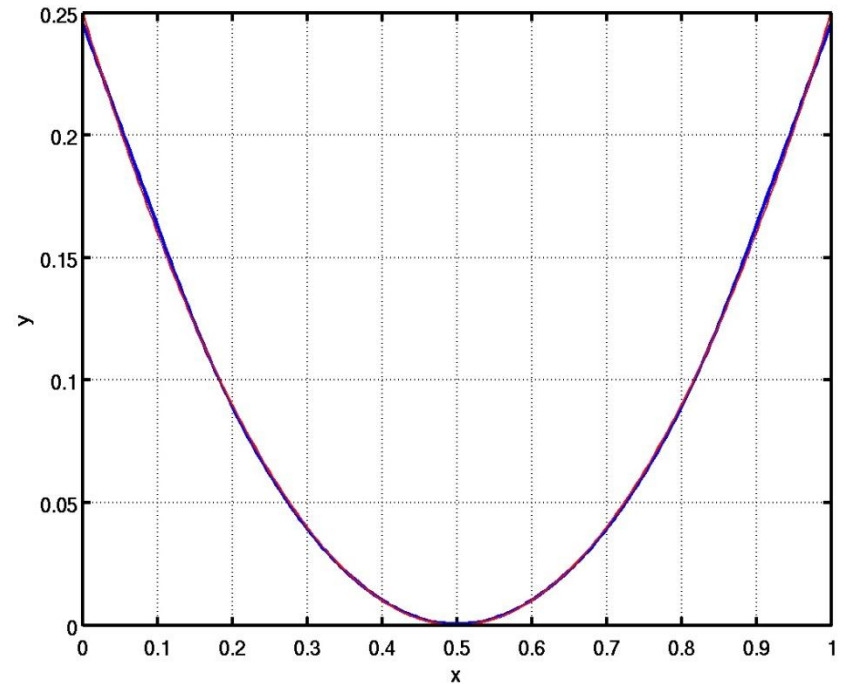
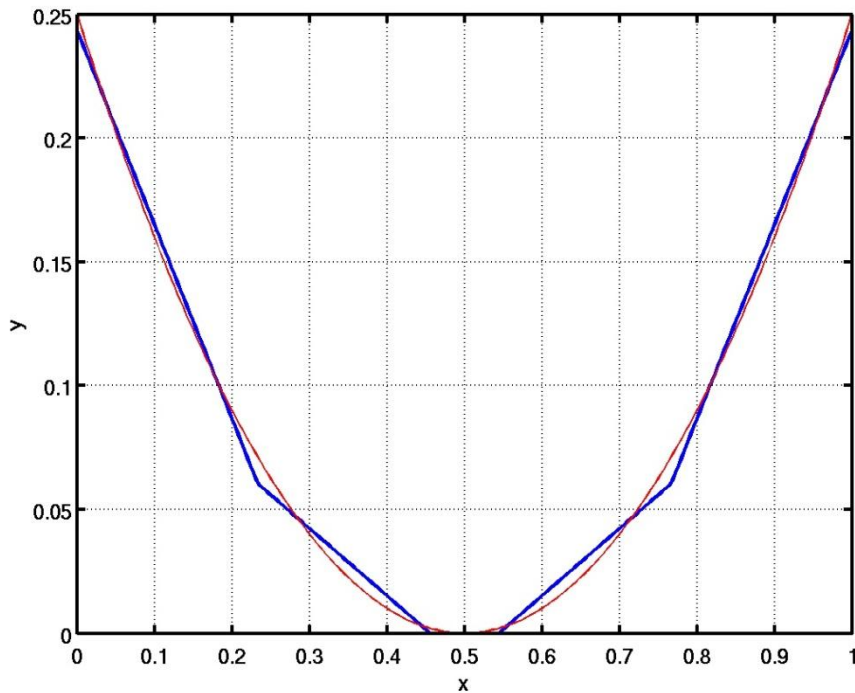
- **The standard deviation of the i-th coefficient is the square root of the (i,i)-th entry of cov(B).**
- **Higher order polynomials can be constructed similarly.**

Adaptive Regression Spline Analysis



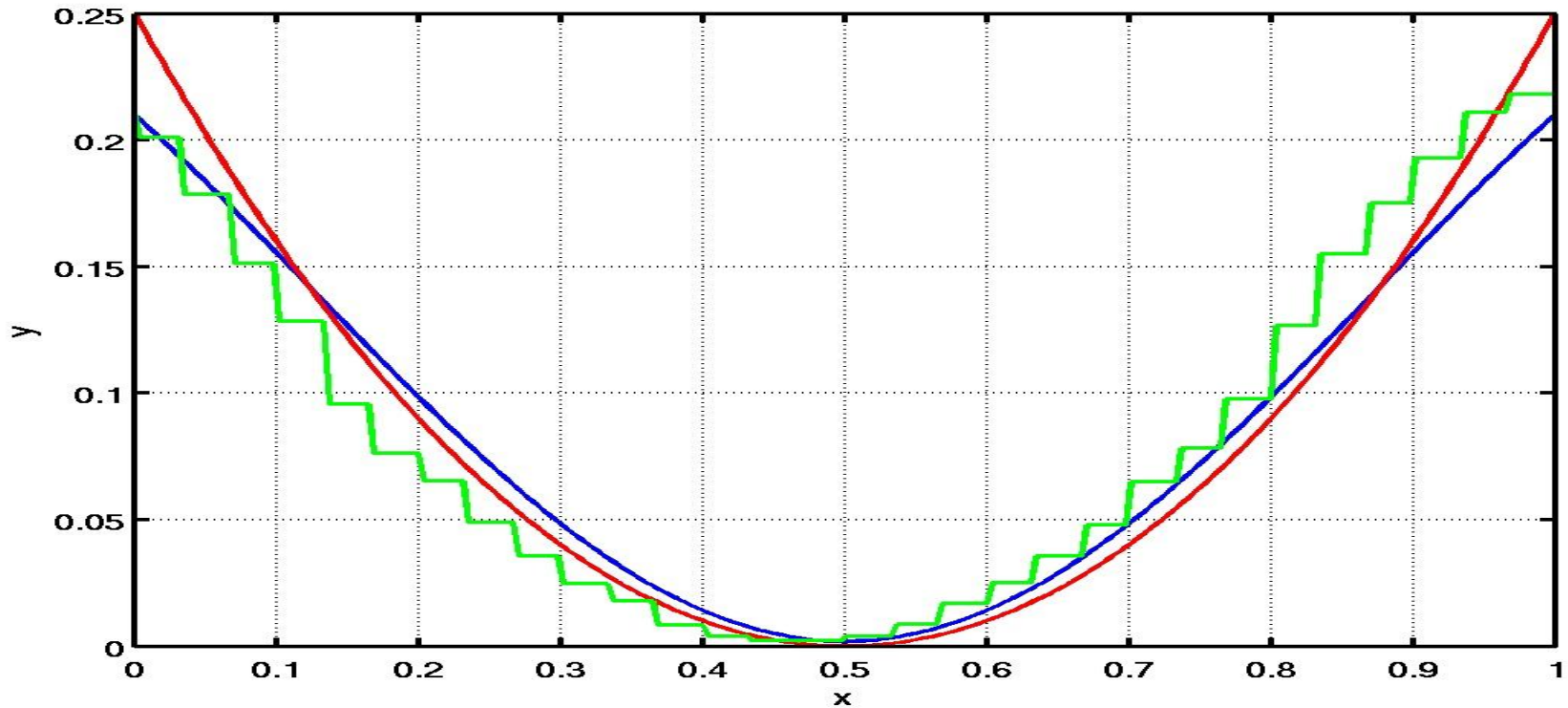
- Function $Y = (X - 0.5) * (X - 0.5)$, $0 \leq X \leq 1$
- Sample size = 21, use equally spaced points
- Left: linear splines; right: cubic splines
- Knots at 0.35, 0.5, 0.65

Adaptive Regression Spline Analysis



- **Function $Y = (X - 0.5) * (X - 0.5)$, $0 \leq X \leq 1$**
- **Sample size = 31, use equally spaced points**
- **Left: linear splines; right: cubic splines**
- **Knots at 0.2333, 0.3667, 0.5, 0.6333, 0.7667**

Other response surface interpolation schemes



- Function $Y = (X - 0.5) * (X - 0.5)$, $0 \leq X \leq 1$
- Sample size = 31, use equally spaced points
- **Red: true curve**, **Blue: Gaussian process**, **Green: sum-of-trees**



Choice of error measures

Definition: Evaluate $S = \{ (X^i, Y^i), i=1, \dots, N, X_i \in \mathbb{R}^m, Y^i \in \mathbb{R} \}$.
Find $f \in F$ (hypothesis function space) such that $V(S, f)$ (some error measure) is minimized.

- **R-square or adjusted R-squares (polynomial regression)**
- **Taylor expansion (truncation error)**
- **Convergence of the function mean (classical learning)**
- **Chi-square (training error, cannot account for generalization error)**
- **Holdout data set (training and test set)**
- **k-fold cross validation (check generalization error)**
- **Statistics on point-wise standard deviation for Gaussian process**
- **Extrapolation analysis: Gower distance**



K-fold Cross validation

- **Given a sample of N points** $S = \{ (X^i, Y^i), i=1, \dots, N, X_i \in \mathbb{R}^m, Y^i \in \mathbb{R} \}$
- **Divide the sample into k roughly same size groups**
- **For $i = 1$ to k**
 - **take out group i and use the rest to build a response surface**
 - **use the response surface to predict the outputs of group i**
 - **compute the sum of squares of the output discrepancies**
- **Add up all k sum of squares, divide by N and assess sufficiency**
- **Advantage: all N sample points are used in the response surfaces**
- **Provide some checking for extrapolation accuracy**
- **Exhaustive cross validation: using $k = N, N/2, N/3, \dots$**
- **Ideal error statistics: approximately Gaussian with zero mean and small standard deviation**



Plan

- **Introduction to uncertainty quantification (UQ)**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**



What is sensitivity analysis (SA)?

- Sensitivity Analysis is the study of how the variation in the output of a model can be apportioned, qualitatively or quantitatively, to different sources of variation.
- It is thus the natural next step after the output uncertainties have been quantified.
- It can be classified into 3 groups:
 - Local sensitivity analysis
 - Screening (qualitative SA, covered previously)
 - Global sensitivity analysis



Recall the Bungee Jump example: use local SA to identify important parameters

Use local sensitivity analysis at the nominal point (50, 70.5, 30)

$$h = H_0 - (2Mg)/(k\sigma)$$

$$\frac{\partial h}{\partial H_0} = 1$$

$$\frac{\partial h}{\partial M} = - \frac{2g}{k\sigma} \Big|_{\sigma=30} = - \frac{4}{9}$$

$$\frac{\partial h}{\partial \sigma} = \frac{2Mg}{k\sigma^2} \Big|_{\sigma=30, M=70.5} = \frac{28}{27}$$

Conclusion: order of importance = $\sigma > H_0 > M$

This conclusion may not be valid if the model is highly nonlinear



Global Sensitivity Analysis is more suitable for multi-physics applications

- **Local Sensitivity Analysis**
 - Computing partial derivatives of output w.r.t input parameters over a small range
- **Global Sensitivity Analysis**
 - Including the influence of scale and shape (nonlinearities, wide range)
 - Evaluating the individual effect while all other factors are varying (complex interactions)



Classical Sensitivity Analysis

- Pearson Correlation Coefficient
 - Linear correlation coefficient assumes linear relationships between inputs and outputs
- Spearman Coefficient
 - It gives a crude measure of nonlinear relationships
- Nonlinear regression analysis
 - It requires the assumption of a specific functional form
- Also, for large number of inputs, these methods require a large number of calculations to get accurate results.

→ Variance decomposition-based methods



The foundation of variance decomposition is the Sobol' property

- Any function can be decomposed into terms of increasing dimensionality, i.e. (such that the mean of each term is 0.)

$$F(x_1, \dots, x_k) = \sum_{i=1}^k F_i(x_i) + \sum_{i=1}^k \sum_{j>i}^k F_{ij}(x_i, x_j) + \dots + F_{1\dots k}(x_1, \dots, x_k)$$

- Then, the total variance is the sum of the variances of the individual terms.

$$V = \sum_{i=1}^k V_i + \sum_{i=1}^k \sum_{j>i}^k V_{ij} + \dots + V_{1\dots k}$$

- This holds true only for functions with uncorrelated inputs (the joint probability distribution function is 0)



We need to define a few sensitivity measures

- Sensitivity index for input I (main effect or 1st order)

$$S_i = \frac{V_i}{V}$$

- Sensitivity index for input i and j (second order)

$$S_{ij} = \frac{V_{ij}}{V}$$

- Total sensitivity index for input i $S_{T_i} = \sum \text{all } V\text{'s involving } i$



Another useful property from statistics

- Variance decomposition based on conditioning input i

$$V = V[E(Y | X_i)] + E[V(Y | X_i)]$$



Variance of conditional expectation
(conditioned on input i)



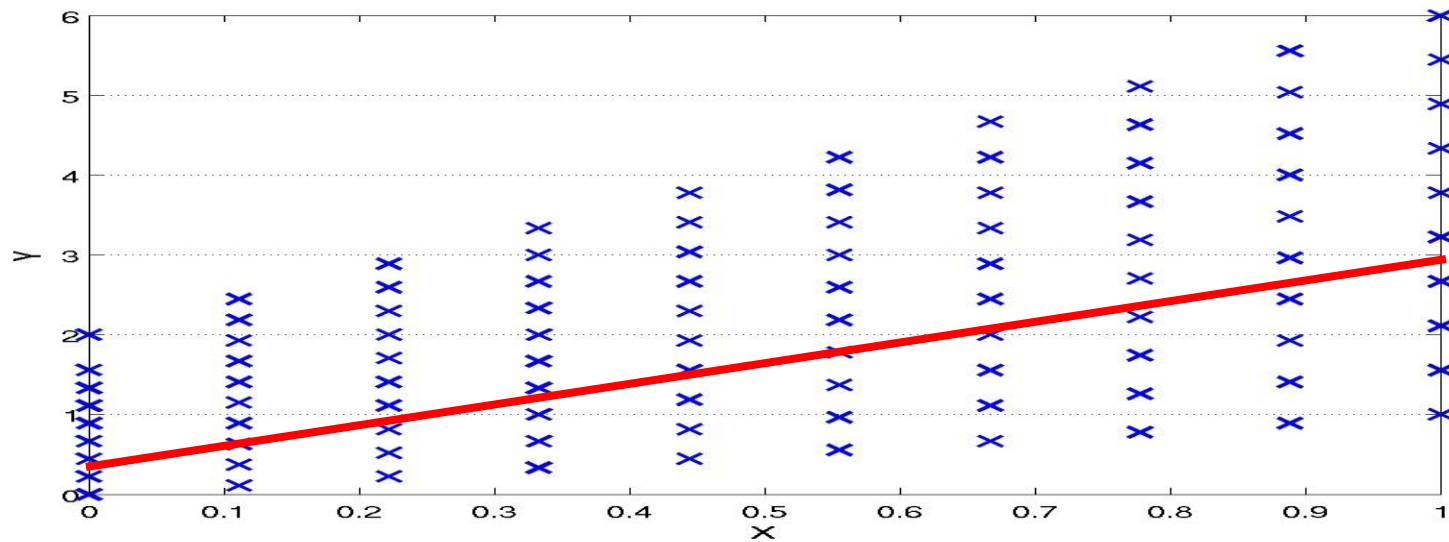
Remaining variability due to
other inputs

- Sensitivity index for input i

$$S_i = \frac{V_i}{V} = \frac{V[E(Y | X_i)]}{V}$$

A pictorial view of variance decomposition

- Given a scatter plot of output with respect to input i



$$V[E(Y | X_i)]$$

Variance of the means (the red line)
The variance of the trend shows the importance of X.

$$E[V(Y | X)]$$

Each column shows the distribution of Y given a fixed X. Calculate the variances and take the mean of all X's



Similarly, we can derive interaction and total sensitivity indices

- interaction study (need different sampling methods)
 - use replicated orthogonal array design

$$V = V[E(Y | X_i, X_j)] + E[V(Y | X_i, X_j)]$$

- total sensitivity indices
 - with correlated inputs, these are better measures
 - can use Fourier Amplitude Sampling Test (FAST) design

$$V = V[E(Y | X_{-i})] + E[V(Y | X_{-i})]$$

$$S_{T_i} = E[V(Y | X_{-i})] / V(Y)$$



Variance Decomposition: An Example

- Given:

$$Y = F(x_1, x_2) = x_1 + 2x_2 + 3x_1x_2$$

where $x_1, x_2 \in [0,1]$ and uniformly distributed

- Basic statistics:

mean $\bar{Y} = 2.25$


variance $\sigma^2 = 1.604$

Apply Sobol' decomposition to the example

1. Rewrite: (so that the mean of each term is 0.)

$$Y = \frac{5}{2} \left(x_1 - \frac{1}{2}\right) + \frac{7}{2} \left(x_2 - \frac{1}{2}\right) + 3 \left(x_1 - \frac{1}{2}\right) \left(x_2 - \frac{1}{2}\right) + \frac{9}{4}$$

2. Calculate the variance of each term:


mean

$$V = V_1 + V_2 + V_{12} = \frac{25}{48} + \frac{49}{48} + \frac{1}{16}$$

3. hence,

Main effect of $x_1 = 25/48$

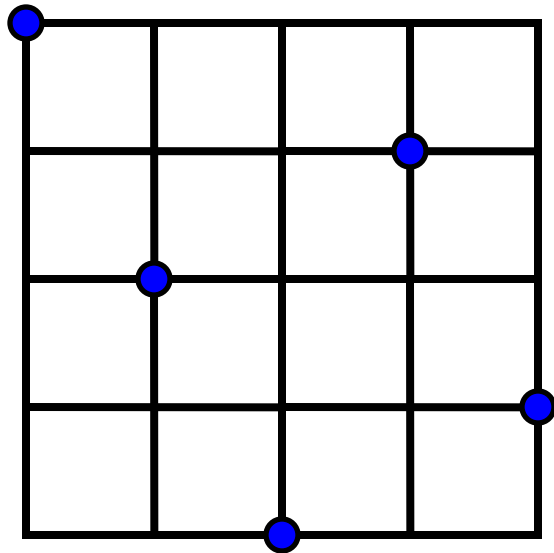
Main effect of $x_2 = 49/48$

interaction (1,2) = $1/16$

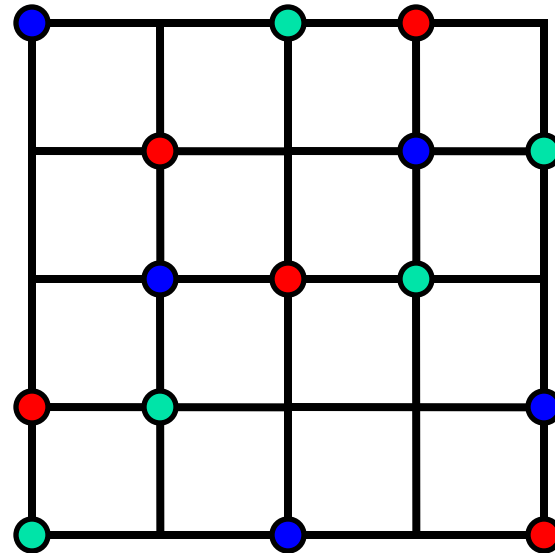
total sensitivity of $x_1 = 25/48 + 1/16 = 7/12$

How to compute
them in practice?

Replicated hypercube offers an efficient way to compute 1st order sensitivity indices



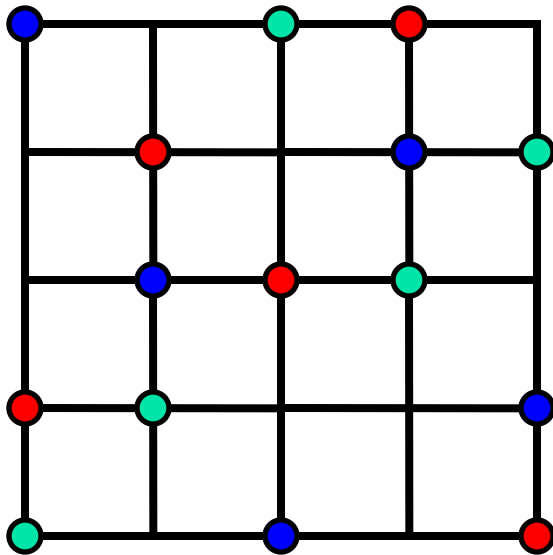
Latin hypercube
(stratified in each dimension)



- Replication 1
- Replication 2
- Replication 3

For each input, there are 3 data per level

How to compute the sensitivity indices?



•

•

•

The same sample can be used for all inputs.



$$\bar{Y}_1 = \frac{1}{R} \sum_{i=1}^R Y_{1i}$$

$$\bar{Y}_2 = \frac{1}{R} \sum_{i=1}^R Y_{2i}$$

.....

$$\bar{Y}_5 = \frac{1}{R} \sum_{i=1}^R Y_{5i}$$

$$\sigma^2 = \frac{1}{R} \sum_{i=1}^R (\bar{Y}_i - \bar{Y})^2$$

How to compute first order sensitivity indices: an example

```
# PSUADE input file (psuade.in)
PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0 1
  variable 2 X2 = 0 1
END
OUTPUT
  dimension = 1
  variable 1 Y
END
METHOD
  sampling = LH
  num_samples = 5000
  num_replications = 100
END
BEGIN APPLICATION
  driver = ./simulator
END
ANALYSIS
  analyzer method = MainEffect
END
END
```

```
/* simulator : pseudocode */

read X1, X2 from input file

 $Y = X1 + 2 * X2 + 3 * X1 * X2$ 

Write Y to output file
```

```
PSUADE run: running sample, nSamples = 5000
=====> MainEffectAnalyzer: mean =
2.2473e+00
=====> MainEffectAnalyzer: standard deviation =
1.2946e+00
*****
* Main Effect Analysis
-----
* total number of samples = 5000
* number of Inputs = 2
----- McKay's correlation ratio -----
INPUT 1 = 3.28e-01 (raw = 5.50e-01) (true~0.52)
INPUT 2 = 6.36e-01 (raw = 1.07e+00) (true~1.02)
Total VCE = 9.64e-01
```

How to compute total sensitivity indices: an example

```
# PSUADE input file (psuade.in)
PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0 1
  variable 2 X2 = 0 1
END
OUTPUT
  dimension = 1
  variable 1 Y
END
METHOD
  sampling = FAST
  num_samples = 57
END
BEGIN APPLICATION
  driver = ./simulator
END
ANALYSIS
  analyzer method = FAST
END
END
```

```
/* simulator : pseudocode */
read X1, X2 from input file
Y = X1 + 2 * X2 + 3 * X1 * X2
Write Y to output file
```

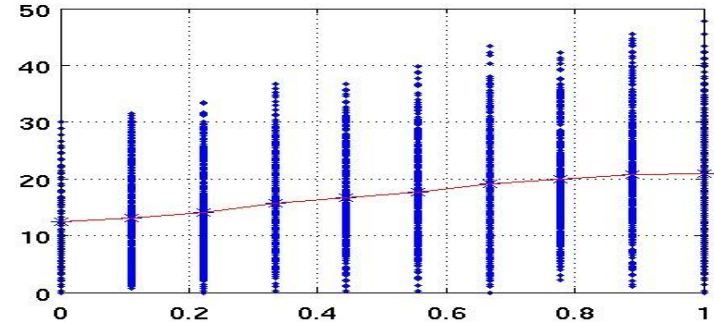
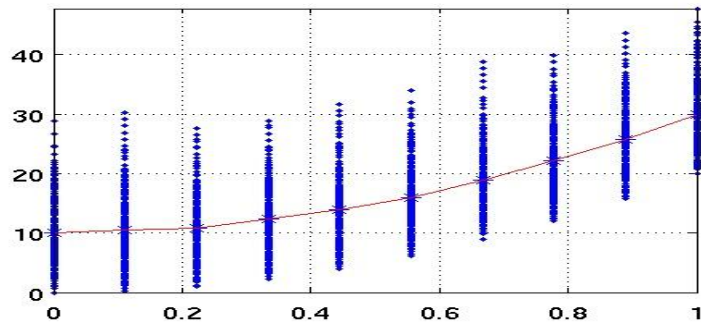
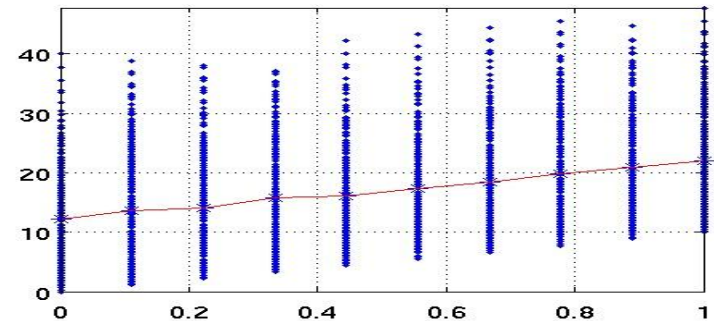
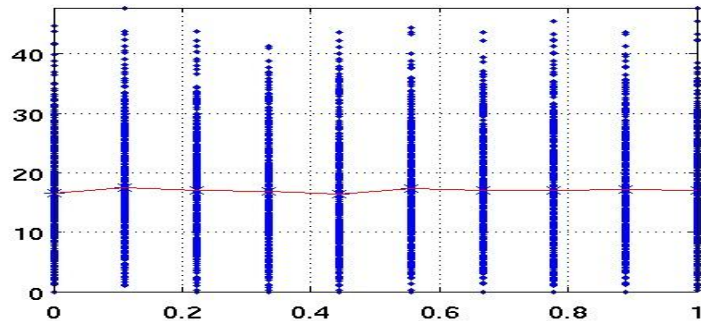
```
=====
* Fourier Amplitude Sampling Test coefficients
-----
* M = 4
* Input 1 = 3.211396e-01 (true ~ 0.3636)
* Input 2 = 6.305783e-01 (true ~ 0.6753)
* Sum of FAST coefficients = 9.517179e-01
* FAST variance = 1.624837e+00
=====
```



Variance decomposition: Another example

- Given: $Y = x_2 + 2x_3^2 + 3x_4x_5$ with 5 inputs
- Use replicated LH with $N=2000$, and $R=50$
- main effect (first order sensitivity indices):
 - input 1: 0.005
 - input 2: 0.09
 - input 3: 0.41
 - input 4: 0.22
 - input 5: 0.22
 - total = 0.95
- total VCE < 1 due to interactions

The scatter plots shows importance and interactions



- the trends of the means show importance and nonlinearities
- the uneven envelopes show interaction (does not show how)



What PSUADE offers for sensitivity analysis

- MOAT design and analysis (qualitative)
- Plackett-Burman design for linear problems (qualitative)
- Delta Test, Sum-of-trees test, MARS test (qualitative)
- Correlation analysis for linear problems
- Ranked correlation analysis for monotonic problems
- Main effect based on replicated Latin hypercube
- Interaction effect based on replicated orthogonal array
- Total effect based on the FAST method
- Main and interaction effects in non-hypercube space
- Main effect and interaction and total sensitivity analysis
- Several other methods: random balanced design (RBD)
- Various graphical analysis tools

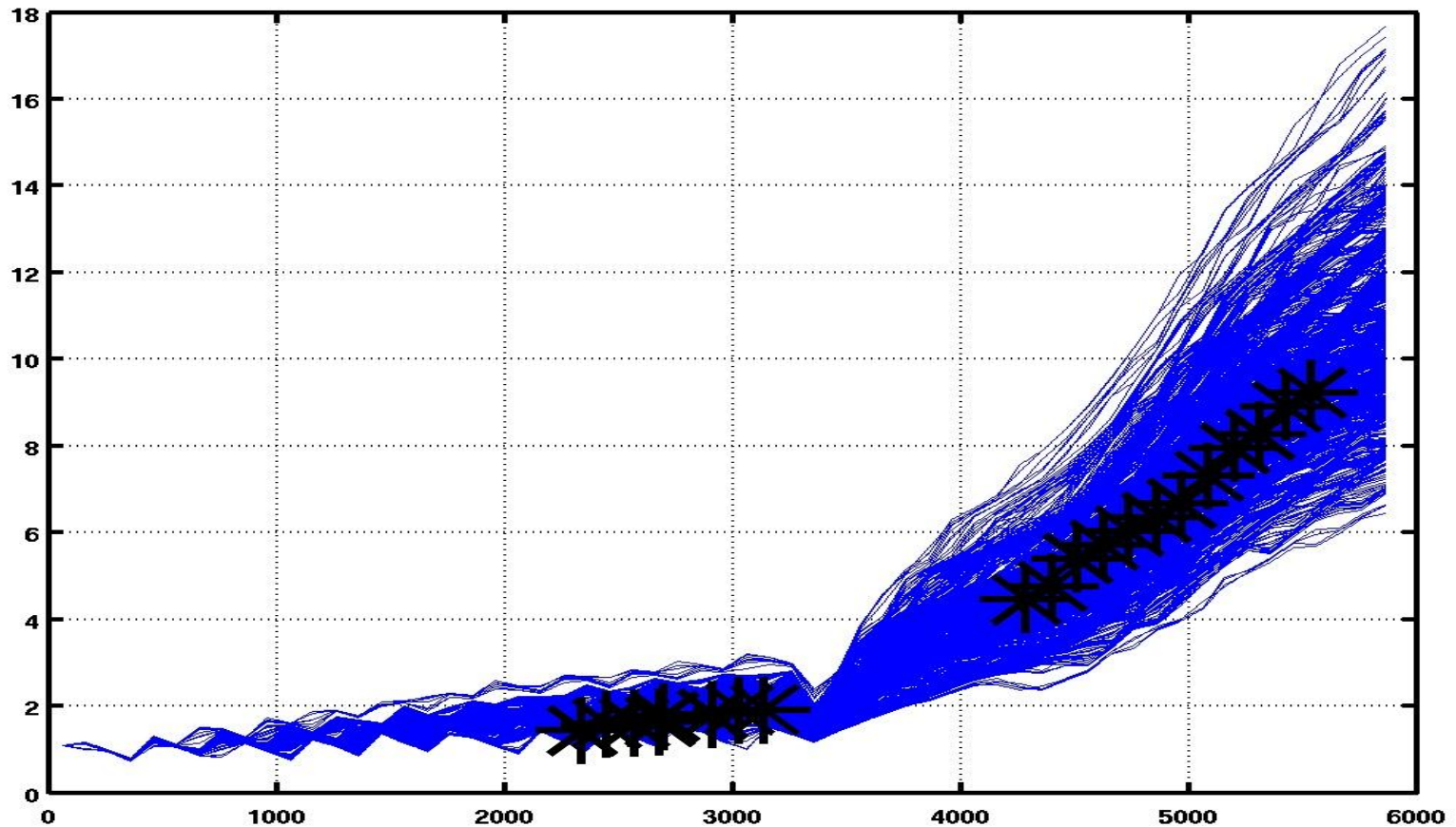


Plan

- **Introduction to uncertainty quantification (UQ)**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**

Why numerical optimization?

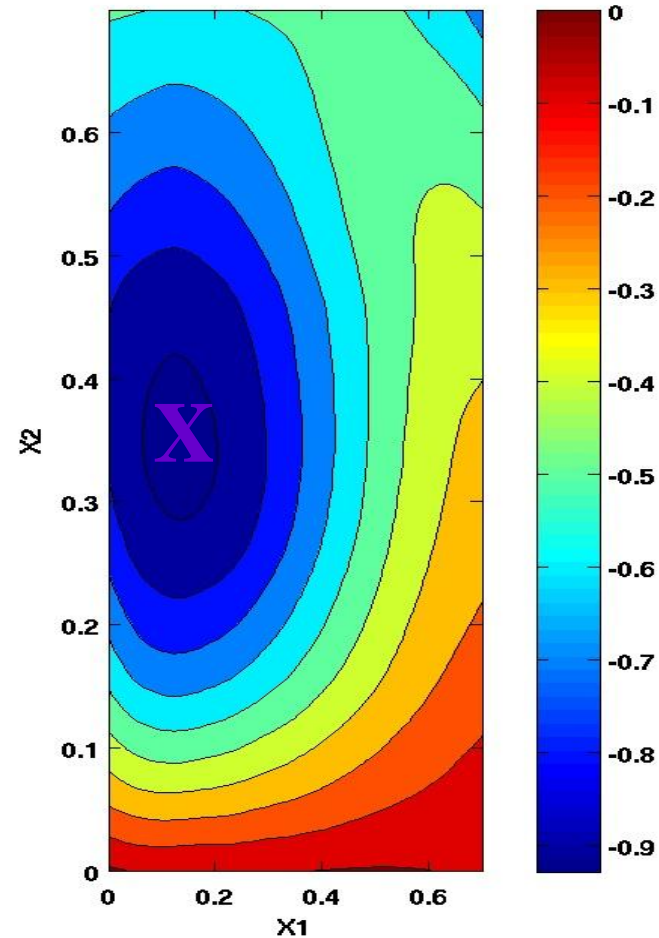
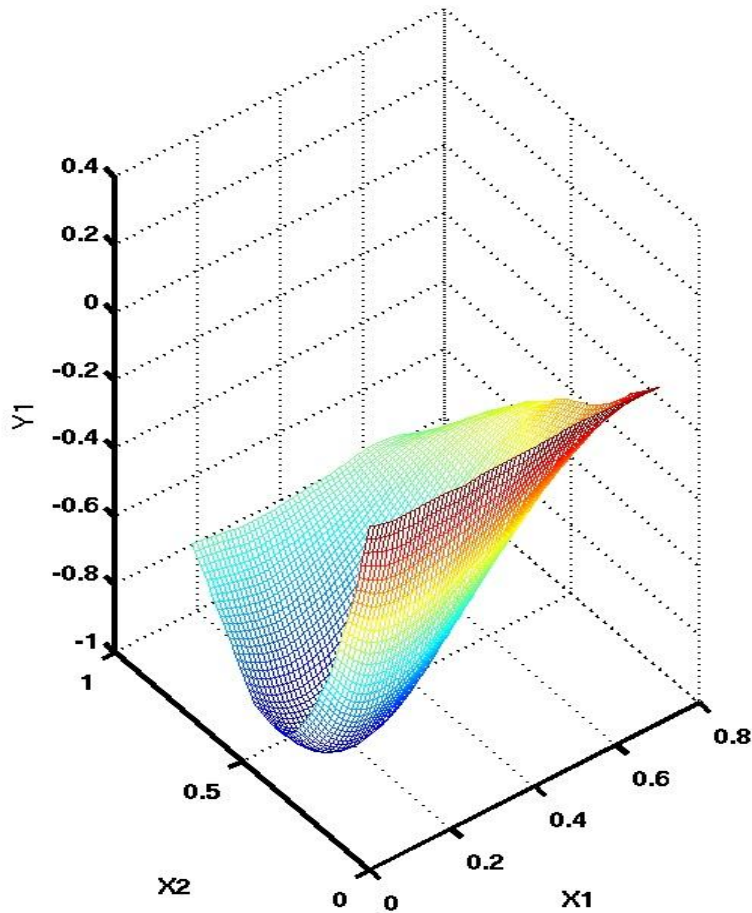
- Model calibration – finding the best match



1000 runs varying all 7 parameters (where '*' are experimental data)

Why numerical optimization?

- Optimal design – finding the best configuration





Model calibration – 2 approaches

- Based on deterministic optimization
 - formulate an objective function (e.g. least-squares)
 - define independent variables and bounds
 - define any inequality constraints
 - run optimization algorithms
- Stochastic optimization (e.g. Bayesian)
 - given data and standard deviation (assume normal)
 - define a likelihood function
 - define independent variables and distributions
 - run Markov Chain Monte Carlo algorithm
- For efficiency reason, response surface is preferred.



Model calibration – deterministic

- formulate an objective function (e.g. least-squares)
- define independent variables and bounds

$$G(\mathbf{X}) = \min_{\mathbf{X}} \sum_{i=1}^n [(Y_i^s(\mathbf{X}) - Y_i^e) / \sigma_i]^2$$

subject to $l_i \leq X_i \leq u_i$

- run optimization algorithm to identify candidates
- if outputs have uncertainties, perform sensitivity analysis in the neighborhood of the candidates



Model calibration – deterministic II

- An example:

4 data points (X,Y): (0,0), (1/3,2/9), (2/3,5/9), (1,1)

- Fit the data into the quadratic form

$$Y = a X + B X^2$$

- Formulate objective function

$$G(X) = \min_{[a,b]} \sum_{i=1}^n [(a X_i + b X_i^2 - Y_i) / \sigma_i]^2$$

subject to $0 \leq a \leq 1$; $0 \leq b \leq 1$

- Use the optimal solution, perform an uncertainty analysis by varying the output data.

Model calibration – deterministic III

```
# PSUADE input file (psuade.in)
PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0 1
  variable 2 X2 = 0 1
END
OUTPUT
  dimension = 1
  variable 1 Y
END
METHOD
  sampling = MC (random guess)
  num_samples = 1
END
BEGIN APPLICATION
  driver = ./simulator
  opt_driver = ./simulator
END
ANALYSIS
  optimization method = cobyla
  optimization max_feval = 10000
  optimization tolerance = 1.0e-6
END
```

```
PSUADE run: jobs completed = 1(out of 1)
*****
PSUADE OPTIMIZATION 1 (1) :
  starting X( 1) = 4.04772655e-01
  starting X( 2) = 5.06000343e-01
  starting Y = 1.26259981e-02
  Cobyla number of function evaluations = 568
  optimum X( 1) = 5.00007850e-01
  optimum X( 2) = 4.99991126e-01
  optimum Y = 5.37008696e-12
*****
PSUADE Optimization Results.
PSUADE Optimization : local optima 1 (1) -
  X 1 = 5.00007850e-01
  X 2 = 4.99991126e-01
  Ymin = 5.37008696e-12
#####
PSUADE OPTIMIZATION : CURRENT GLOBAL
MINIMUM -
  X 1 = 5.00007850e-01
  X 2 = 4.99991126e-01
  Ymin = 5.37008696e-12
#####
PSUADE Optimization : desired minimum found.
```



Bayesian calibration

- An example:

4 data points (X, Y) : $(0,0)$, $(1/3, 2/9)$, $(2/3, 5/9)$, $(1,1)$

- Fit the data into the quadratic form

$$Y = a X + b X^2$$

- Formulate likelihood function

$$L(X | Y) = \exp\left\{-0.5 \sum_{i=1}^n [(a X_i + b X_i^2 - Y_i) / \sigma_i]^2\right\}$$

a and $b \in \text{uniform}[0,1]$

- Run Markov Chain Monte Carlo to get the posterior distribution (not just the optimum)



Model calibration – deterministic III

Input script for creating
response surface

```
# PSUADE input file
PSUADE
INPUT
  dimension = 2
  variable 1 X1 = 0 1
  variable 2 X2 = 0 1
END
OUTPUT
  dimension = 1
  variable 1 Y
END
METHOD
  sampling = LH
  num_samples = 200
END
BEGIN APPLICATION
  driver = ./simulator
  opt_driver = ./simulator
END
END
```

```
[linux %] psuade
psuade> load psRS
psuade> rsmcmc
rsmcmc: perform MCMC on response surfaces
MCMC Burn-in nSamples (default) = 10000
MCMC maximum nSamples (default) = 10000
MCMC number of bins (default) = 20
*****
Enter the standard deviation (> 0) : 0.01
*****
MCMC INFO: creating response surfaces.
MCMC Phase 1: burn in
 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
MCMC Phase 2: create posterior (10.)
 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
MCMC: input 1 statistics (mean,stdev) = 5.100836e-01 3.210077e-03
MCMC: input 2 statistics (mean,stdev) = 5.091668e-01 4.284858e-03
MCMC: input 1 at peak = 5.125624e-01
MCMC: input 2 at peak = 5.125125e-01
MCMC: output 1 at peak = -3.114816e-03
MCMC: likelihood at peak = 1.411236e+07
MCMC: final matlabmcmc.m file has been created.
MCMC: matlabmcmc2.m file (2-input analysis) is ready.
psuade> quit
[linux %]
```



Plan

- **Introduction to uncertainty quantification (UQ)**
- **Uncertainty analysis and sampling**
- **Dimension reduction (screening)**
- **Response surface analysis**
- **Sensitivity analysis**
- **Model calibration/design optimization**
- **Model validation**



Rigorous validation increases credibility of the model

- **What is validation?**
 - “The determination of the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.” – AIAA 1998
- **Hierarchical validation (bottom-up)**
 - **Unit (one physics, e.g. hydrodynamics)**
 - **Subsystem (coupled physics, e.g. hydrodynamics+materials)**
 - **Full system**
- **Validation methods**
 - **Conceptual validation (qualitative)**
 - **Data validation (quantitative)**

Models are imperfect realizations of physical processes

- Physical process

$$B(\Phi; X_1, \dots, X_n) + \varepsilon_b$$

Measurement errors

- Computer model

$$\tilde{B}(\Psi \in \tilde{\Phi}; \tilde{X}_1, \dots, \tilde{X}_n; \delta_\varepsilon)$$

Approximate coupling

Experimental data (with errors)

$$\tilde{Y} \approx Y$$

- scalar
- time series
- images
- time images

$$\{X_1, \dots, X_n\} \& \{\tilde{X}_1, \dots, \tilde{X}_n\}$$

Exact and approximate design parameters

$$\Phi = \{\Phi_1, \dots, \Phi_m\}$$

Exact physics models

$$\tilde{\Phi} = \{\tilde{\Phi}_1(V_1), \dots, \tilde{\Phi}_m(V_m)\}$$

Inexact physics models (uncertainties parametrized)



Aspects of Model Validation

- **Conceptual model validation**
 - **Determine that the model representation is reasonable for the intended application of the model (e.g. laminar/turbulent flow)**
 - **Determine that the model responses are consistent with the physical phenomena (e.g. linearity, interactions)**
 - **Sensitivity analysis and design exploration methods can aid in conceptual validation**
- **Data validation (need uncertainty/sensitivity analyses)**
 - **Use historical data and new experiments to ensure the model is sufficiently accurate for the intended application of the model:**
 - **Need validation metrics: physical quantities + distance metrics**
 - **Apply at various levels: unit, subsystem, system, benchmarks**
 - **Acceptance: hypothesis tests or statistical tests**
- **Documentation (the procedure and results)**
 - **To communicate the progress of validation**



Validation Data and Metrics

- Scalar quantities (with probabilistic distribution from UQ)
 - comparison of 2 probability distribution functions
 - Parametric or nonparametric: e.g. chi-squared statistics, Kolmogorov Smirnov statistics, Bhattacharyya distance, ...
 - Should select appropriate measures for a given application
- Function (vector) quantities (time series)
 - Apply transforms with respect to some basis functions
 - e.g. wavelet basis, orthogonal polynomials, PCA
 - Gather statistics and apply scalar metrics
- Images (2D or 3D)
 - Apply transforms to condense the information
 - e.g. scan line transforms, surface to volume ratio, shapelet
 - Gather statistics and apply scalar metrics



A Methodology for computing validation metric

- Decide on a diagnostic variable and compile the data set ($\{D_i, \sigma_{D_i}\}$)
- To create output distribution for the computer model
 - Identify the set of uncertain parameters and their ranges
 - If too many parameters, do a down-select to get a few parameters
 - Create response surface for the subset of parameters
 - Sampling (small) on the response surface to compute ($\{Y_i, \sigma_{Y_i}\}$)
- Use chi-square formula to compute distance metric

$$\text{Dist}(\{D_i\}, \{Y_i\}) = \sum_{i=1}^m \frac{(Y_i - D_i)^2}{\sigma_{D_i}^2 + \sigma_{Y_i}^2}$$

- Use hypothesis testing to determine acceptance
- Next: use deterministic/Bayesian optimization to calibrate parameters



Distance and Metric Space

- A distance function on a given set M is a function $d: \Omega \times \Omega \rightarrow \mathbb{R}$,
 - Ω is a m -dimensional probability space
 - \mathbb{R} is a real number
- that satisfies the following conditions (Let X, Y be both in Ω):
 - $D(X, Y) \geq 0$, and $D(X, Y) = 0$ if and only if $X=Y$.
 - It is symmetric: $D(X, Y) = D(Y, X)$.
 - It satisfies the triangle inequality: $D(X, Z) \leq D(X, Y) + D(Y, Z)$.
- An example: Euclidean distance between 2 points in Ω

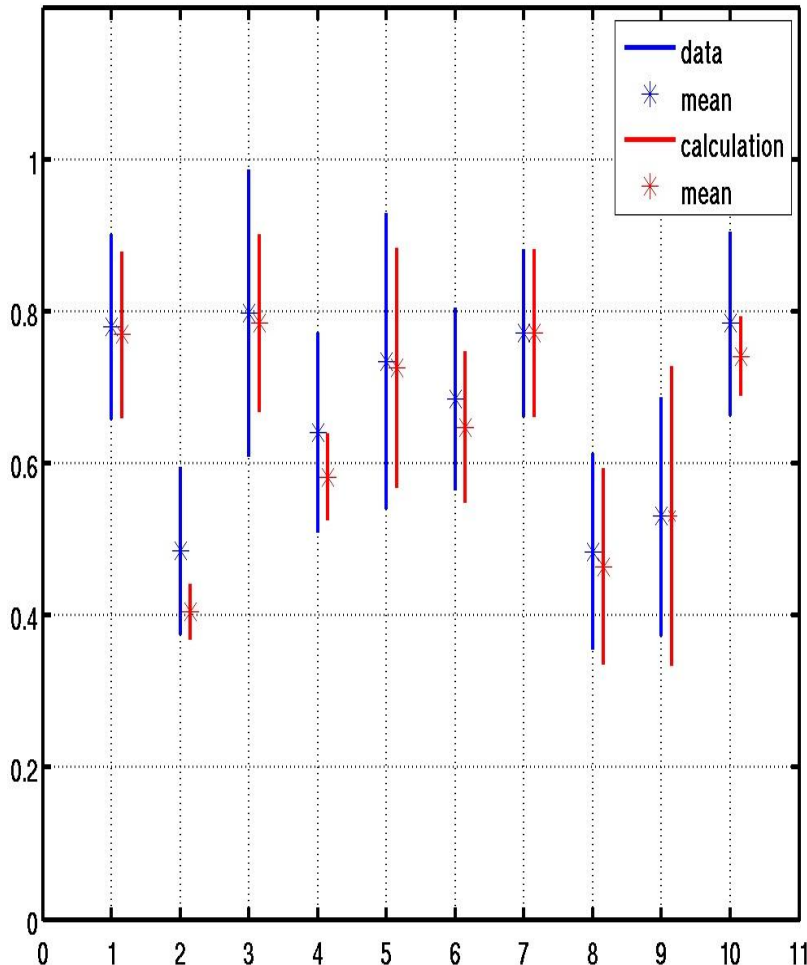
$$D_E(X, Y) = \left(\sum_{i=1}^m |X_i - Y_i|^2 \right)^{1/2}$$



Distance Metric can either be Parametric or Non-parametric

- Non-parametric: m-D probability distribution (or histogram if discrete)
 - correlation built in to the probability distribution
 - metrics: Chernoff/Bhattacharyya, χ^2 , Kolmogorov-Smirnov
 - Minkowski/Chebyshev distances may be used
 - Other metrics: earth mover distance (EMD), JD
- Parametric (known form of distributions, e.g. Gaussian)
 - Ω reduced to a mean (m-D) vector and a covariance matrix
 - The distance can be classified further as with or without correlation
 - No correlation: e.g. χ^2 , Minkowski, Euclidean
 - With correlation: Hotelling, Mahalanobis ([similar to Cornwall's](#))

Parametric Validation for multiple outputs with no correlation



1a. No calculational uncertainty

- one calculation per data
- each red line is a single point

$$\text{Dist}(\{D_i\}, \{Y_i\}) = \sum_{i=1}^m \frac{(Y_i - D_i)^2}{\sigma_{D_i}^2}$$

1b. Calculational uncertainty assessed

- Some number of calculations to compute model uncertainty

$$\text{Dist}(\{D_i\}, \{Y_i\}) = \sum_{i=1}^m \frac{(Y_i - D_i)^2}{\sigma_{D_i}^2 + \sigma_{Y_i}^2}$$

These are parametric distance metrics



Parametric Validation for multiple outputs with correlation

1. No calculational uncertainties

- 1 calculation per device (Y_i, D_i : mean vectors, Σ_{D_i} : covariance matrix)
- Use Mahalanobis metric

$$\text{Dist}(\{D_i\}, \{Y_i\}) = \sum_{i=1}^m (Y_i - D_i)^T \Sigma_{D_i}^{-1} (Y_i - D_i)$$

2. Parametric calculational uncertainties

- **many** (but still a small number of) calculations
- compare 2 multi-dimensional distributions

$$\text{Dist}(\{D_i\}, \{Y_i\}) = \sum_{i=1}^m (Y_i - D_i)^T (\Sigma_{D_i} + \Sigma_{Y_i})^{-1} (Y_i - D_i)$$

- Caveat: Data covariance matrix may not be readily available
(can assume data and calculations have same correlation)



Nonparametric Validation (skewed/multi-modal)

1. Construct probability distribution for data ($p(D)$): probability density)
2. Use many runs to construct model probability distribution ($p(M)$)
 - To Compare 2 multi-dimensional distributions, use Bhattacharyya distance (Y is a single or multiple output space), or

$$D_B(p(M), p(D)) = -\ln(BC) \quad \text{where } BC = \sqrt{\int_Y p(M|Y)p(D|Y)}$$

- Kolmogorov-Smirnov distance (P : cumulative density function)

$$D_{ks}(P(M), P(D)) = \max_i (|P(M|Y_i) - P(D, Y_i)|)$$

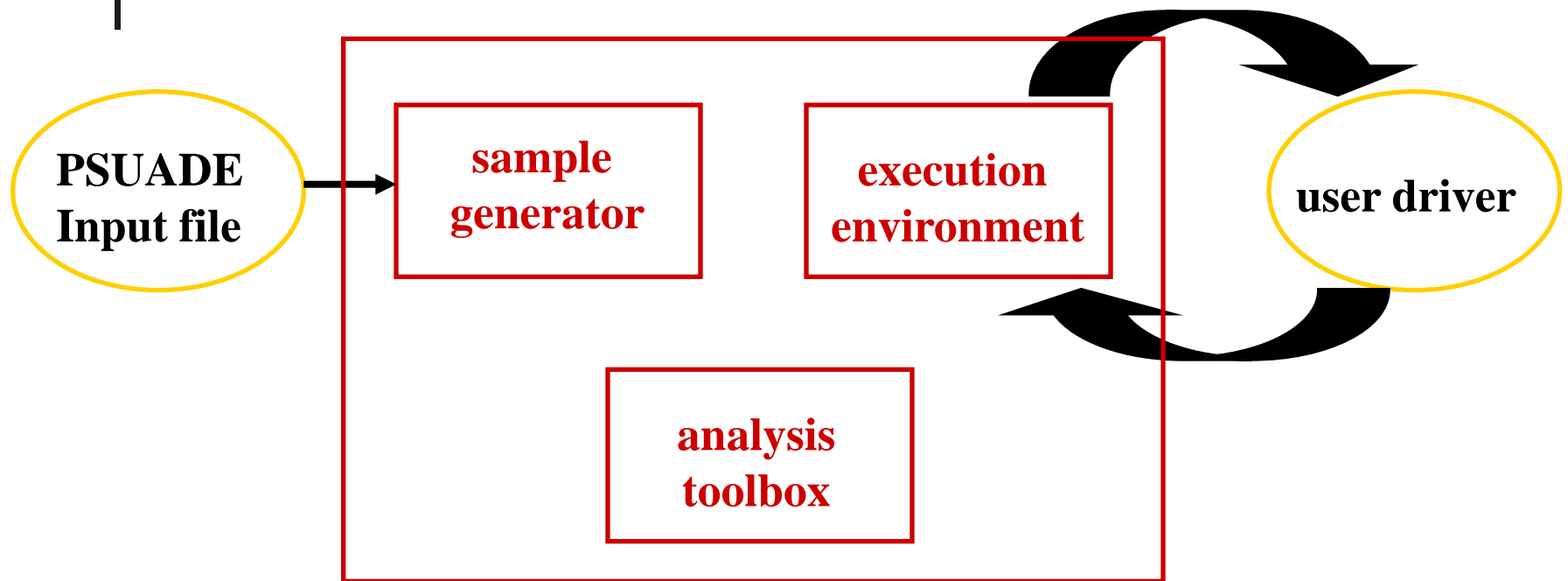
3. Use this metric with the methodology to measure and guide progress



PSUADE

A **P**roblem **S**olving environment for
Uncertainty **A**nalysis and **D**esign
Exploration

PSUADE has 3 major components



1. PSUADE creates the sampling design
2. PSUADE manages the simulation resource allocations
3. PSUADE analyzes the results

For short jobs, the entire process can be fully automated.



Example 1: creating user program

- **The bungee jumping example**

$$H = H_0 - (2Mg)/(k\sigma)$$

$$H_0 \in [40,60]$$

$$M \in [67,74]$$

$$\sigma \in [20,40]$$

$$g = 9.8, k = 1.5$$

- **run: psuade**
- **psuade> gendriver**
- **use C program (option 1)**
- **give a name (e.g. simulator.c)**
- **psuade> quit**
- **add to the C program the above equation**
- **compile to create executable: simulator**



Example 1: creating PSUADE input file

- run psuade again (no argument)
- psuade> **geninputfile**
- give a name (e.g. psuade.in)
- answer all questions
 - number of outputs=1, MC, sample size=1000
 - driver program = simulator
 - maximum number of jobs=1, run time=0
 - deterministic, do not generate data file
- **quit** after the file has been created
- examine your psuade input file



PSUADE: Input and output sections

**PSUADE
INPUT**

dimension = 3

variable 1 X1 = 40.0 60.0

variable 2 X2 = 67.0 74.0

variable 3 X3 = 20.0 40.0

PDF 1 U

PDF 2 N 0.0 1.0

PDF 3 L 5.0 5.0

PDF 4 T 0.0 1.0

END

OUTPUT

dimension = 1

variable 1 Y

END

.

.

END



PSUADE: Method section

PSUADE

•

•

METHOD

sampling = FACT

sampling = FF5

sampling = CCC4

sampling = MOAT

sampling = LH

num_samples = 1000

num_replications = 50

END

•

•

END



PSUADE: Application section

PSUADE

•
•

APPLICATION

driver = ./simulator

driver = ./psuadeData

opt_driver = NONE

max_parallel_jobs = 4

launch_interval = 10

launch_only

gen_inputfile_only

nondeterministic

max_job_wait_time = 10000

save_frequency = 1

END

•
•

END



PSUADE: Analysis section

PSUADE

•
•

ANALYSIS

analyzer method = Moment

analyzer method = MOAT

analyzer method = MainEffect

analyzer output_id = 1

analyzer rstyle = MARS

analyzer rstyle = linear

analyzer rstyle = quadratic

analyzer rstyle = ANN

optimization method = cobyla

other optimization parameters

END

•
•

END



Example 1: running and analyzing

- run: **psuade psuade.in**
- When it is done, do: **mv psuadeData psuadeSave**
- launch psuade again
- psuade> **load psuadeSave**
- psuade> **printlevel 4**
- psuade> **ua** (examine the moments)
- ... respond 'y' to creating an pdf file
- ... give a name (e.g. pdf.m)
- quit psuade
- launch matlab/display to display distribution



Example 1: visualizing

- **run: psuade**
- **psuade> load psuadeSave**
- **psuade> rs2**
 - **Select input 1 and 3 and nominal for others**
 - **Answer 'no' to setting lower/upper bounds**
- **psuade> quit**
- **launch matlab/scilab and run: matlabrs2**
- **you can also try out rs3 to visualize all 3 inputs**



Example 1: using python driver script

- run: **psuade**
- **psuade> gendriver**
 - use option **2**
 - give a file name (e.g. **simulator.py**)
 - ‘**no**’ to dependency, no appl/batch/support files
 - **3** inputs: **H0, M, S**; **1** output
- **psuade> quit**
- edit the **simulator.py** file to add the equation
- run: **chmod 755 simulator.py**
- uncomment the last few lines to clear **workdir.x**
- modify the driver in the **psuade.in** to **simulator.py**
- run: **psuade psuade.in**



Example 2: Morris screening

- Use the Morris test problem: 20 parameters, first 10 are important, the first 6 have interactions, 7 nonlinear
- The user driver can be copied from `/usr/gapps/psuade/Examples/Morris20 (simulator.c)`
- compile `simulator.c` to `simulator` (use `-lm`)
- create `psuade` input file
- now run: `psuade psuade.in`
- create screening diagram
- observe: large means for parameter 1 to 10
- observe: large standard deviations for parameter 1 to 7
- launch `matlab/scilab` and plot the screening diagram



Example 3: Quantitative sensitivity analysis

- Use again the bungee jumping example
- Grab the simulator: `simulator.c`
- Compile `simulator.c` to `simulator` (use `-lm`)
- Create a `psuade.in` file for main effect analysis
(Latin hypercube, `num_samples=5000`, `num_replications=100`)
- Run: `psuade psuade.in`
- Observe: the correlation ratios for the 3 inputs
- launch `matlab/scilab` and plot the scatter plots



PSUADE: Interactive mode

```
[linux %] psuade
```

```
*****
```

```
*** Welcome to PSUADE (version 2.0) ***
```

```
*****
```

```
PSUADE - A Problem Solving Environment for  
          Uncertainty Analysis and Design Exploration
```

```
psuade> load datafile
```

```
psuade> help
```

```
Help topics:
```

```
    io    (file read/write operations)
```

```
    uqsa  (UQ/SA functions)
```

```
    misc  (miscellaneous functions)
```

```
psuade> splot
```

```
matlabsp.m is now available for scatter plot.
```

```
psuade> list
```

```
psuade> me
```

```
...
```

```
psuade> max
```

```
...
```

```
psuade> ua
```

```
...
```



Example 4: numerical optimization

- **Obtain an optimization example simulator**
`psuade/Examples/SandOpt/simulator.c`
- **Examine the simulator and see what it is doing**
- **Compile `simulator.c` to `simulator`**
- **Create a `psuade` input file for creating response surface**
- **Run the simulator and use `psuade` and `matlab/scilab` to view the response surface**
- **Create a `psuade.in` file for optimization (use `cobyla` with random initial guess: MC)**
- **Run the input file several times to observe the minimum.**



Example 5: Bayesian optimization

- Obtain an optimization example simulator from `psuade/Examples/MCMC/simulator.c`
- Examine the simulator and discuss what it is doing
- Compile `simulator.c` to `simulator`
- Create a `psuade` input file for creating response surface
- Run `psuade` in interactive mode
- Load the response surface data file and run: `rsmcmc`
 - figure out what standard deviation to use
- Use `matlab/scilab` to view `XXXmcmc.m` and `XXXmcmc2.m`
- Try running with different standard deviation in `simulator` and `mcmc` runs