École chercheurs MEXICO, La Rochelle, Mars 2018

Introduction to statistical modelling

Nicolas Durrande, nicolas@prowler.io

PROWLER.io, Cambridge (UK) – Mines St-Étienne (France)

Outline of today's lecture

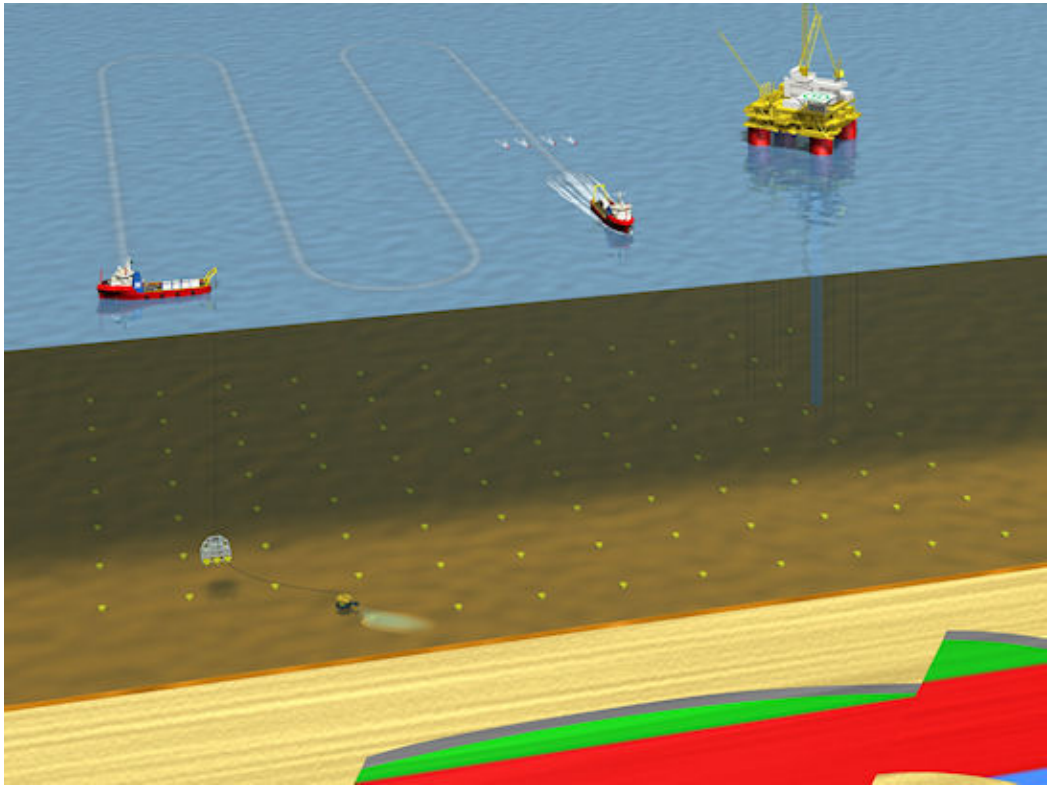
- Introduction to costly to evaluate functions.
- A few words on statistical modelling.
- Design of experiments

Why are statistical models relevant in engineering?

There is a wide variety of situations where getting data about a system performance can be extremely expensive.

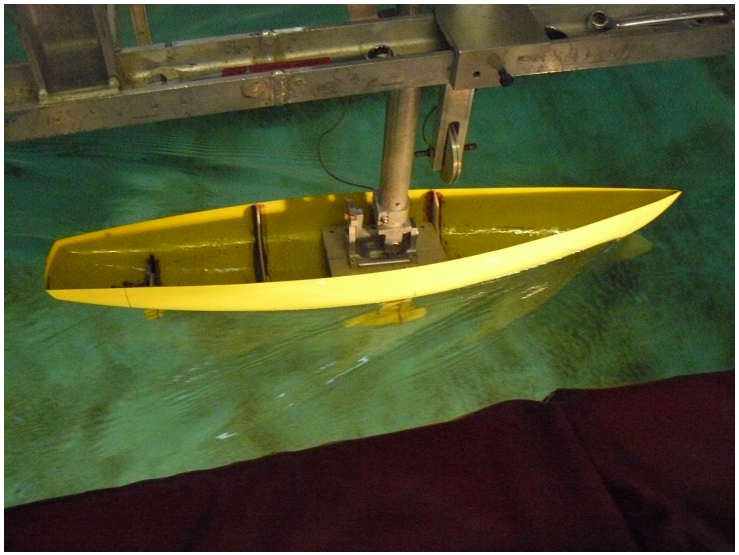
- real world experiments
- destructive tests
- prototyping
- numerical experiments

Example: real world experiments



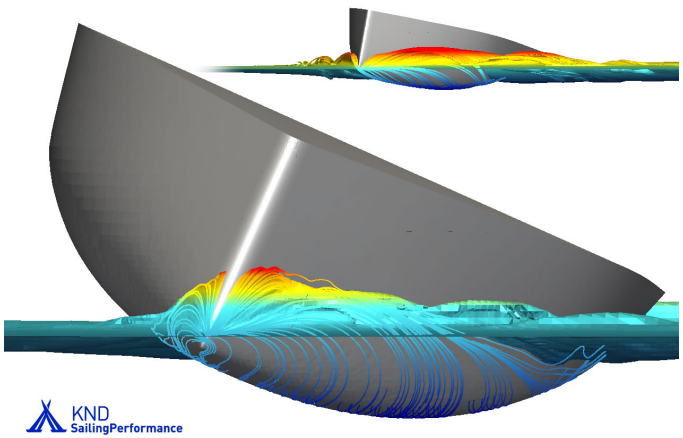
Example: Destructive tests

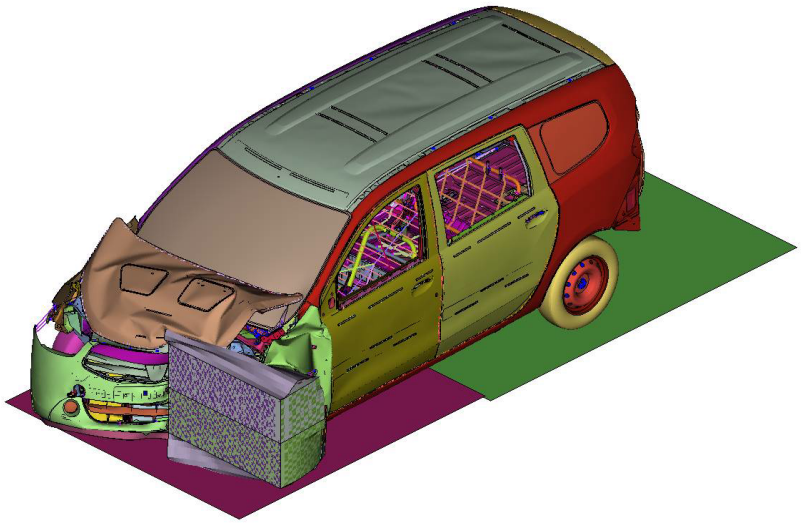
Example: Prototyping of a boat shape



Knowing the drag for a given design requires costly experiments

Example: Numerical experiments





Numerical experiments are less expensive but can be very time consuming!

In all these cases, the variable of interest can be seen as a function of the input parameters

$$y = f(x).$$

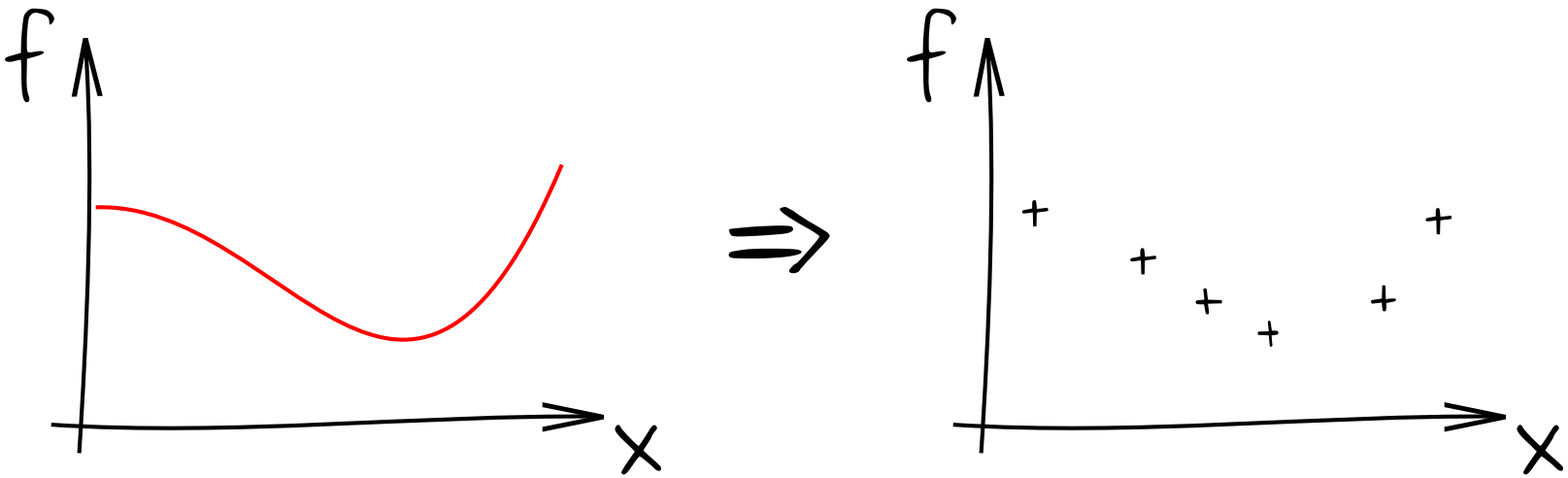
where f is a **costly to evaluate function**.

In the following, we will assume that

- $x \in \mathbb{R}^d$: There are many input parameters
- $y \in \mathbb{R}$: The output is a scalar.

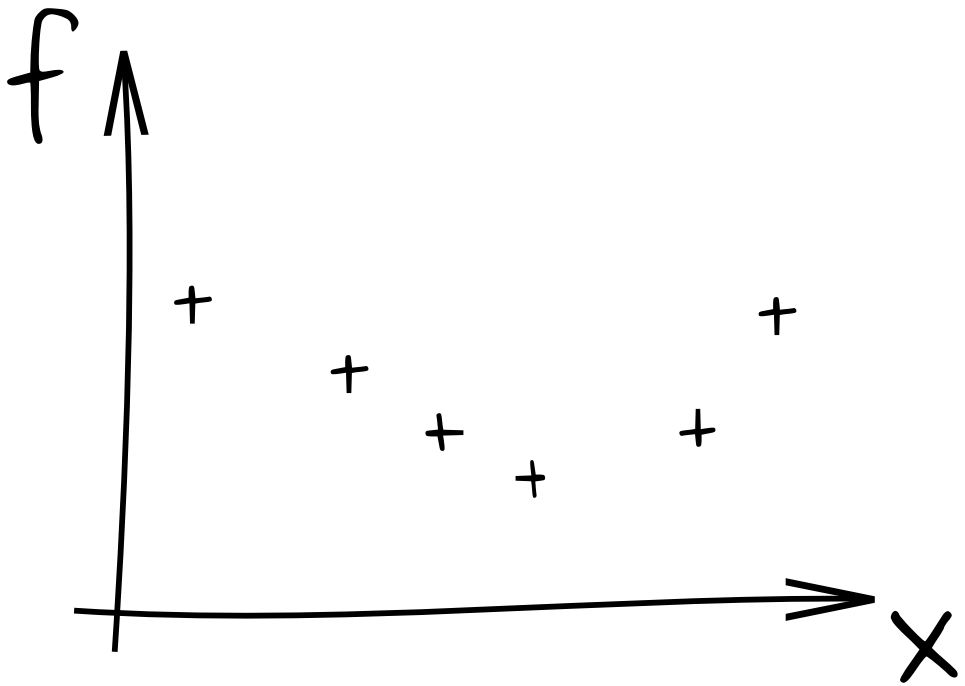
The fact that f is **costly to evaluate** changes a lot of things...

1. Representing the function is not possible...



The fact that f is **costly to evaluate** changes a lot of things...

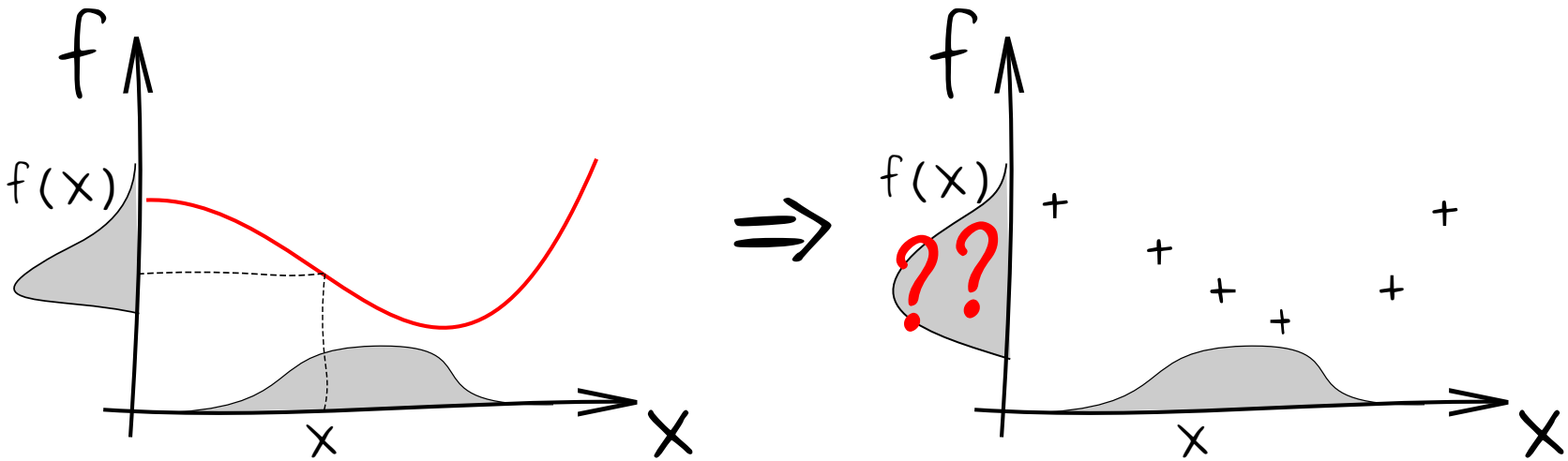
2. Computing integrals is not possible...



What is the mean value of f?

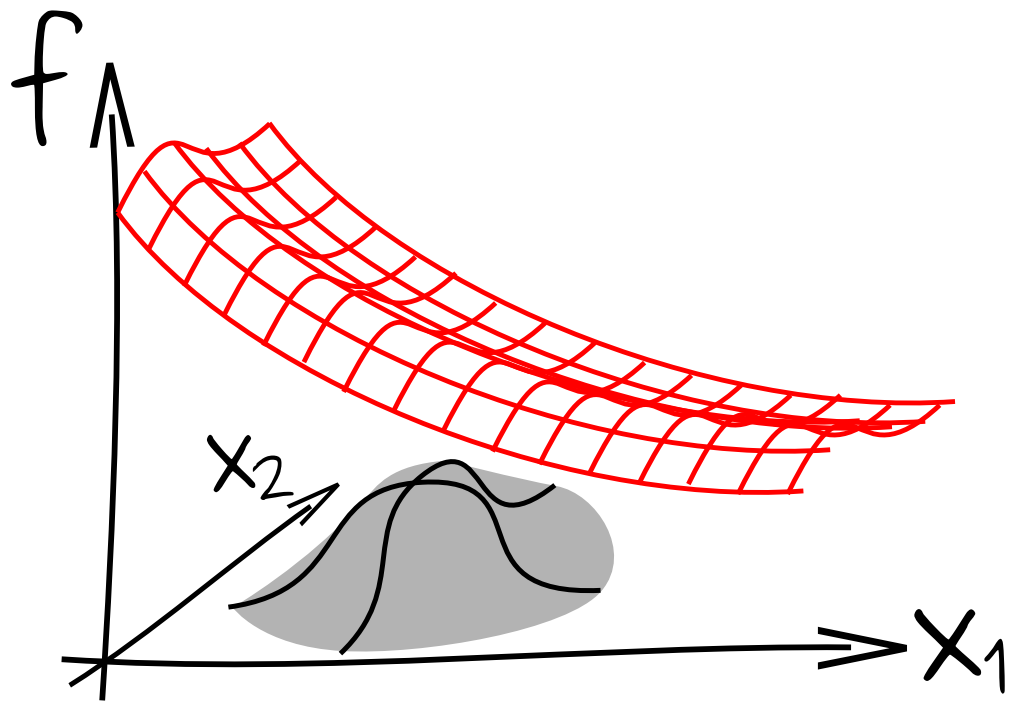
The fact that f is **costly to evaluate** changes a lot of things...

3. Uncertainty propagation is not possible...



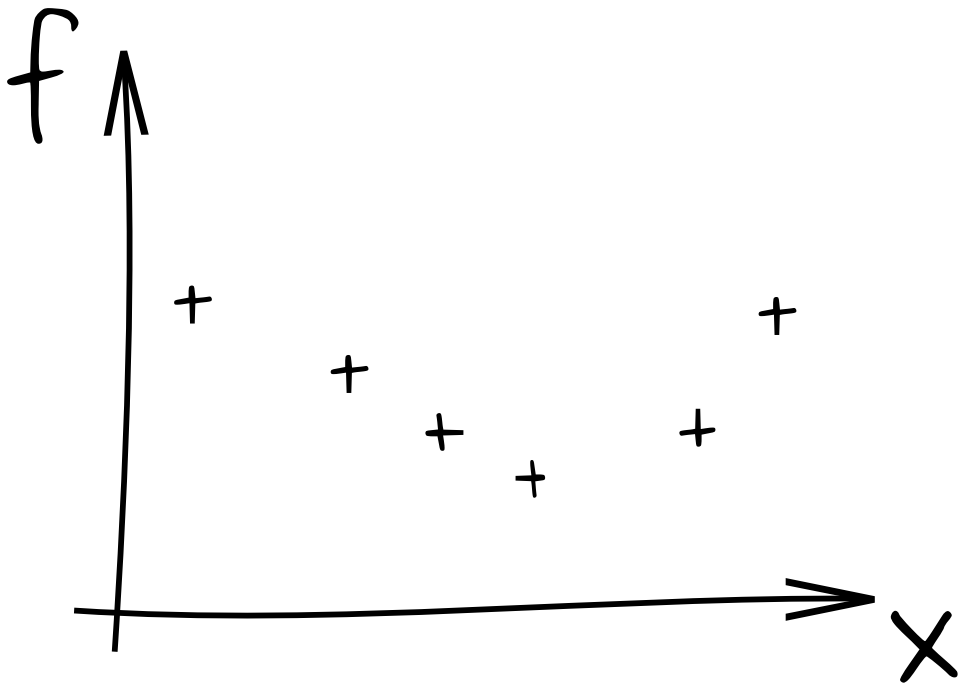
The fact that f is **costly to evaluate** changes a lot of things...

4. Sensitivity analysis is not possible...



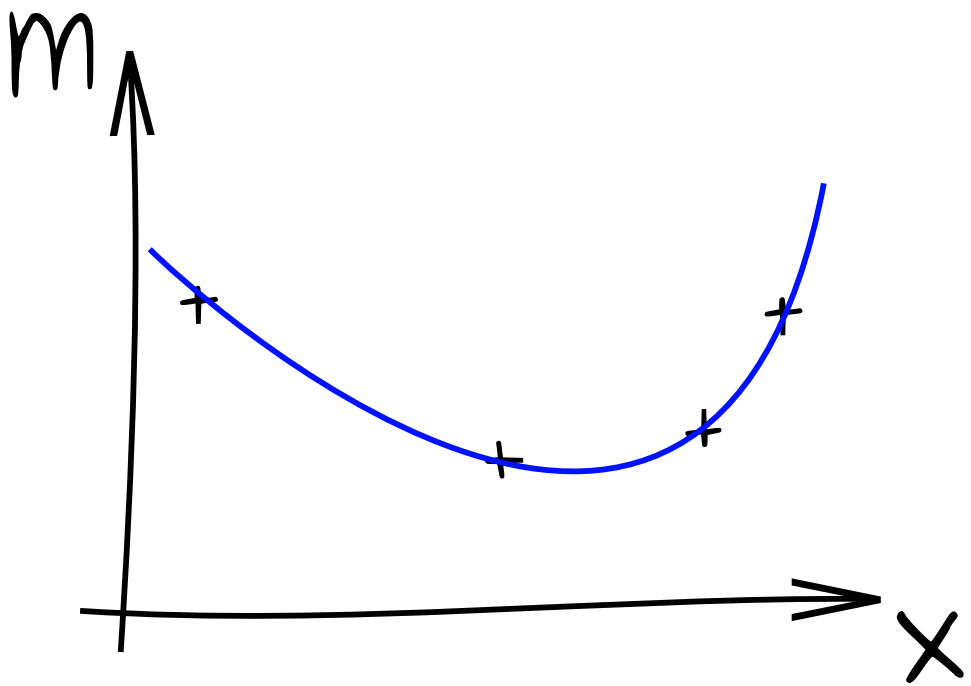
The fact that f is **costly to evaluate** changes a lot of things...

5. Optimisation is also tricky...



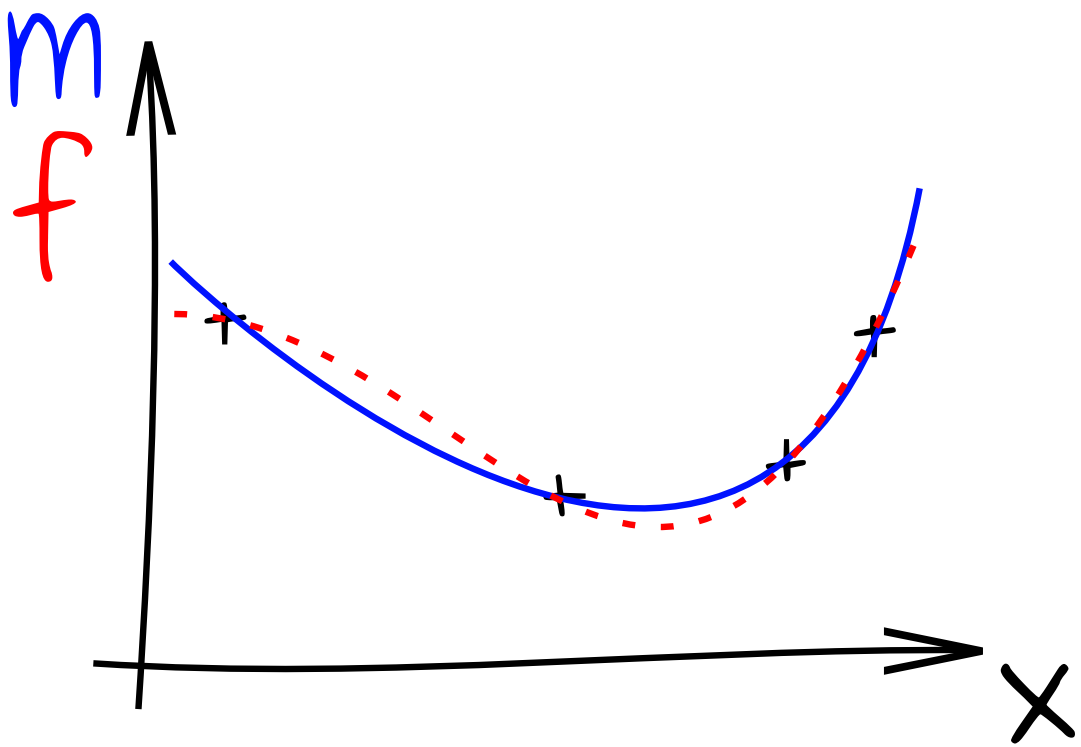
Statistical models

The principle of statistical modelling is to use the data to build a mathematical approximation of the function.

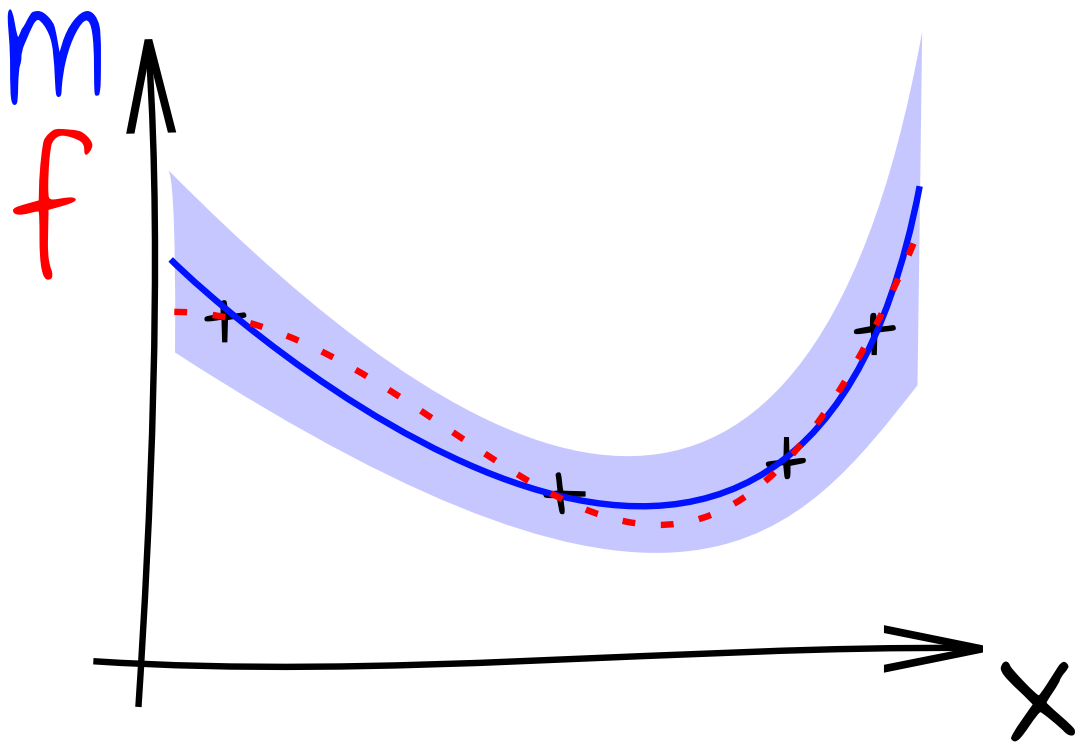


The model can then be used to answer all previous questions

Of course, there is a difference between f and m...



Why **statistical models**?
We want to be able to quantify the model error:



The confidence intervals can be used to obtain a **measure of uncertainty on the value of interest**.

In the sequel, we will use the following notations :

- The set of observation points will be represented by a $n \times d$ matrix $X = (X_1, ..., X_n)^t$
- The vector of observations will be denoted by F : $F_i = f(X_i)$ (or $F = f(X)$).

Before discussing statistical models (tomorrow), we will talk about **design of experiments**

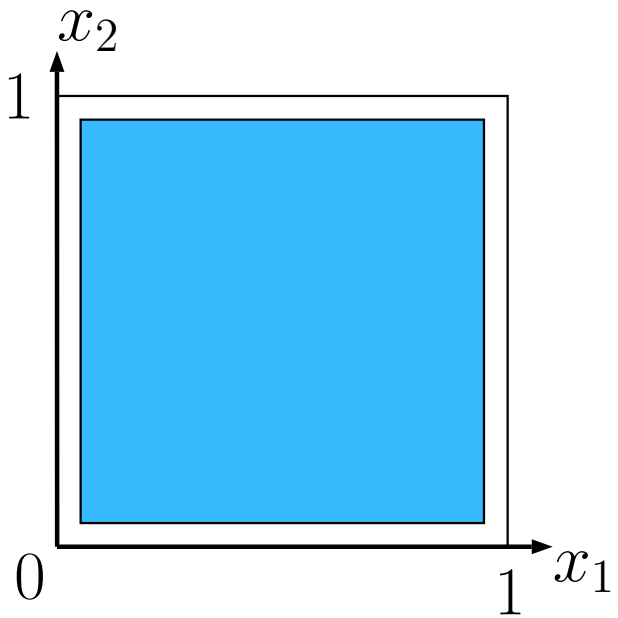
If you have a budget of n function calls, which points of the input space should be evaluated?

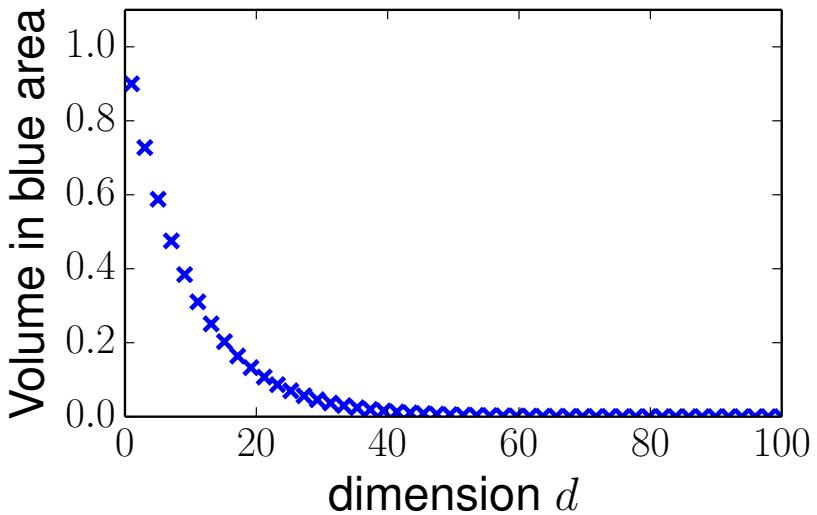
Design of Experiments

Intuition is often misleading in high-dimension:

Examples 1/2

- Points in the unit cube can be far away
  $\rightarrow$ the diagonal of the unit cube is of length $\sqrt{d}$
- All the volume is near the domain boundaries
  $\rightarrow$ let us consider a hypercube of size 0.9 included in the the unit cube:

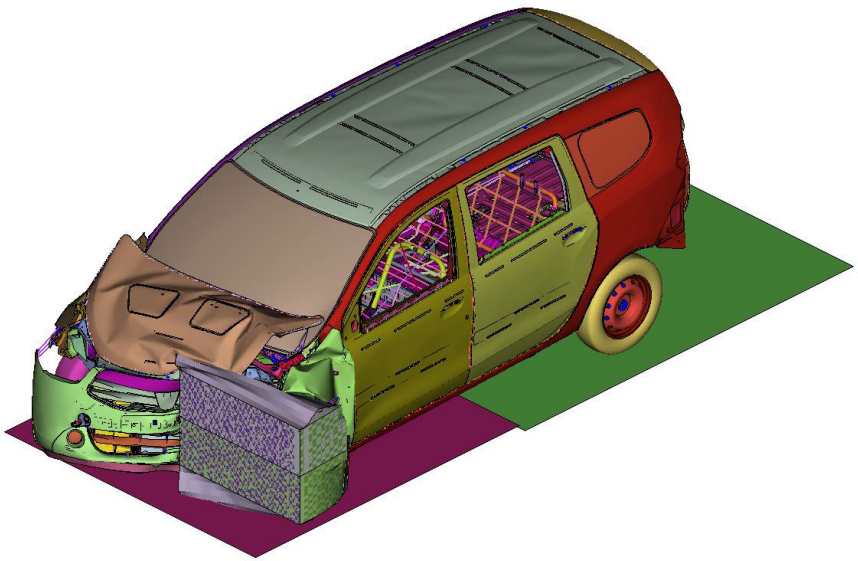




Intuition is often misleading in high-dimension:

Examples 2/2

- The number of vertices of an hypercube increases faster than we usually think

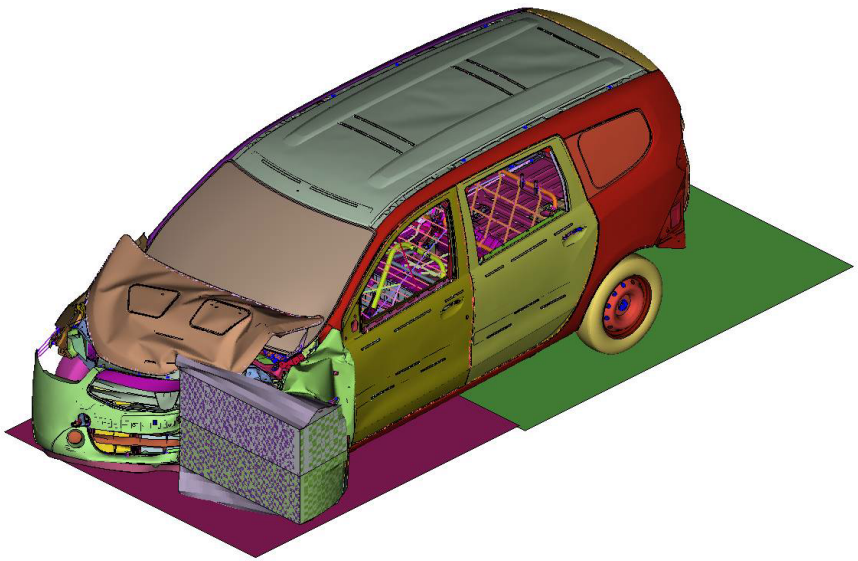


Testing all combinations of min and max input values for the 50 parameters would require...

Intuition is often misleading in high-dimension:

Examples 2/2

- The number of vertices of an hypercube increases faster than we usually think



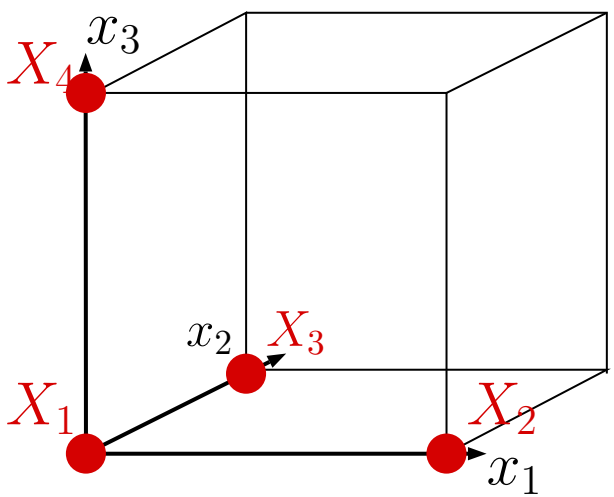
Testing all combinations of min and max input values for the 50 parameters would require...
3000 times the age of the universe!
($d = 50 \rightarrow 2^d \approx 1.e15$)

One at a time design

An intuitive way to check the influence of various variable is to make them change one at the time.

- All variables are fixed at a reference value (0 for example)
- One variable is changed at a time to see if there is an influence

Example



point	x_1	x_2	x_3
X_1	0	0	0
X_2	1	0	0
X_3	0	1	0
X_4	0	0	1

pros and cons of this kind of design:

+ require only $d + 1$ observations
+ are easy to interpret

– they can only see linear effects:

$$m(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

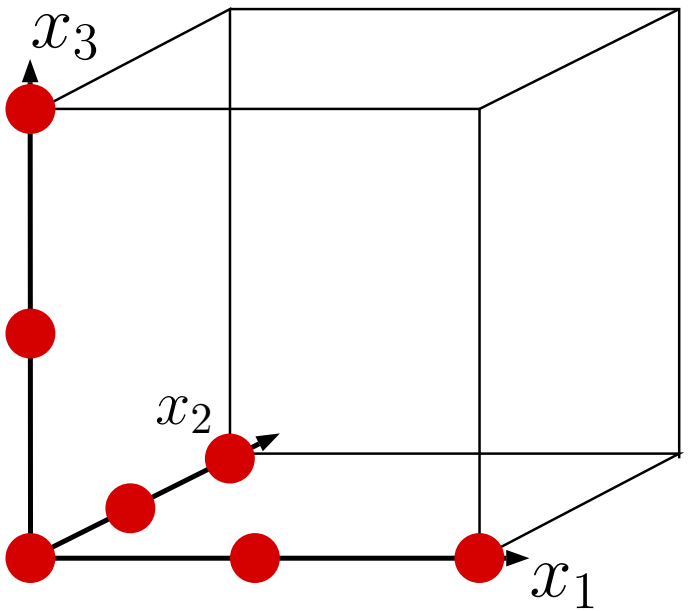
– they do not cover the space

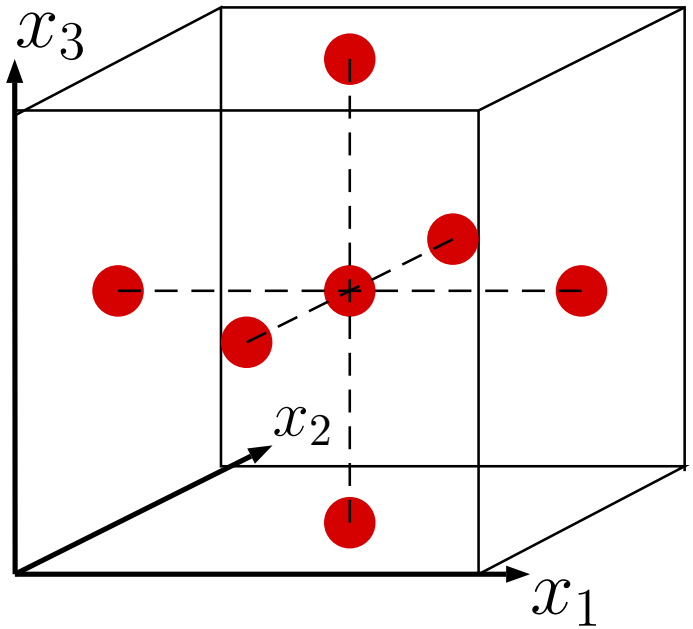
Exercise

How can this kind of design be adapted to estimate quadratic effect?

Solution

Quadratic effects can be estimated with either

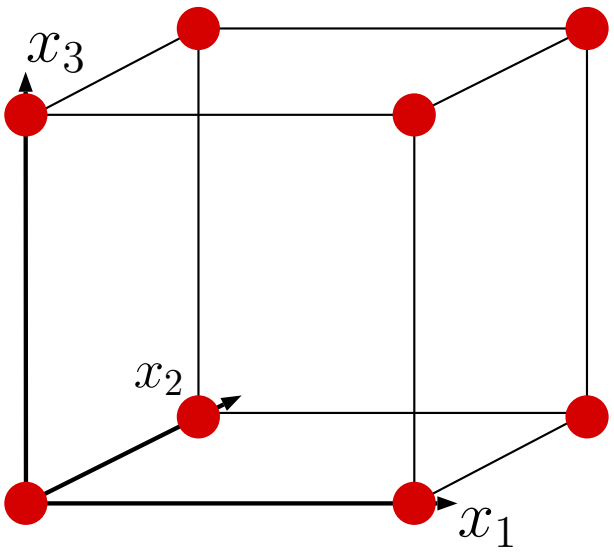




we sometime talk about “star shaped” design.

Factorial designs

The principle of factorial design is to consider all combinations for $x_i \in \{0, 1\}$:



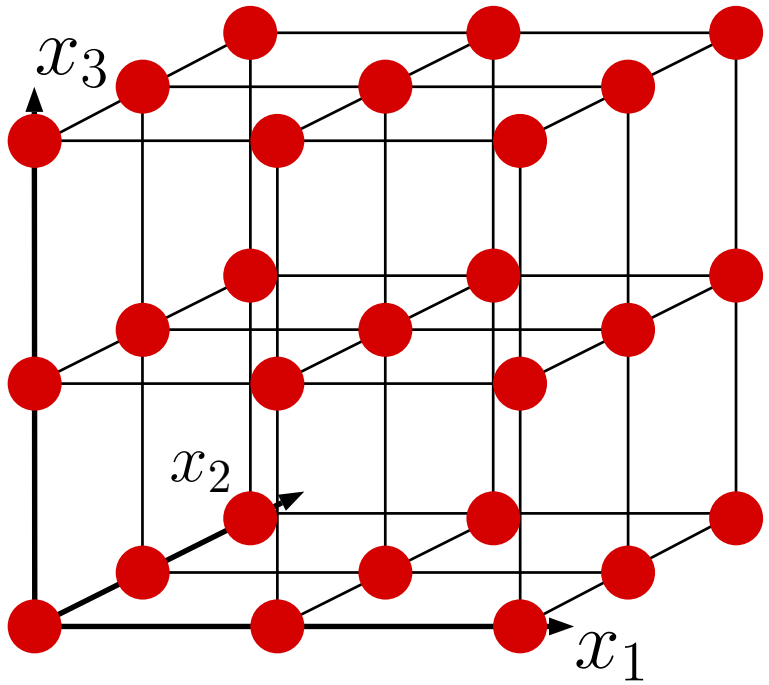
pros They allow to get all interaction terms:

$$\beta_0 + \sum_k \beta_k x_k + \sum_{j,k} \beta_{j,k} x_j x_k + \beta_{1,2,3} x_1 x_2 x_3$$

cons The number of evaluation is unrealistic when d is large

Factorial designs

It is also possible to build factorial designs with k levels:



This allows to compute quadratic effects but the number of evaluations k^d is even less realistic...

Conclusion on classical designs:

pros:

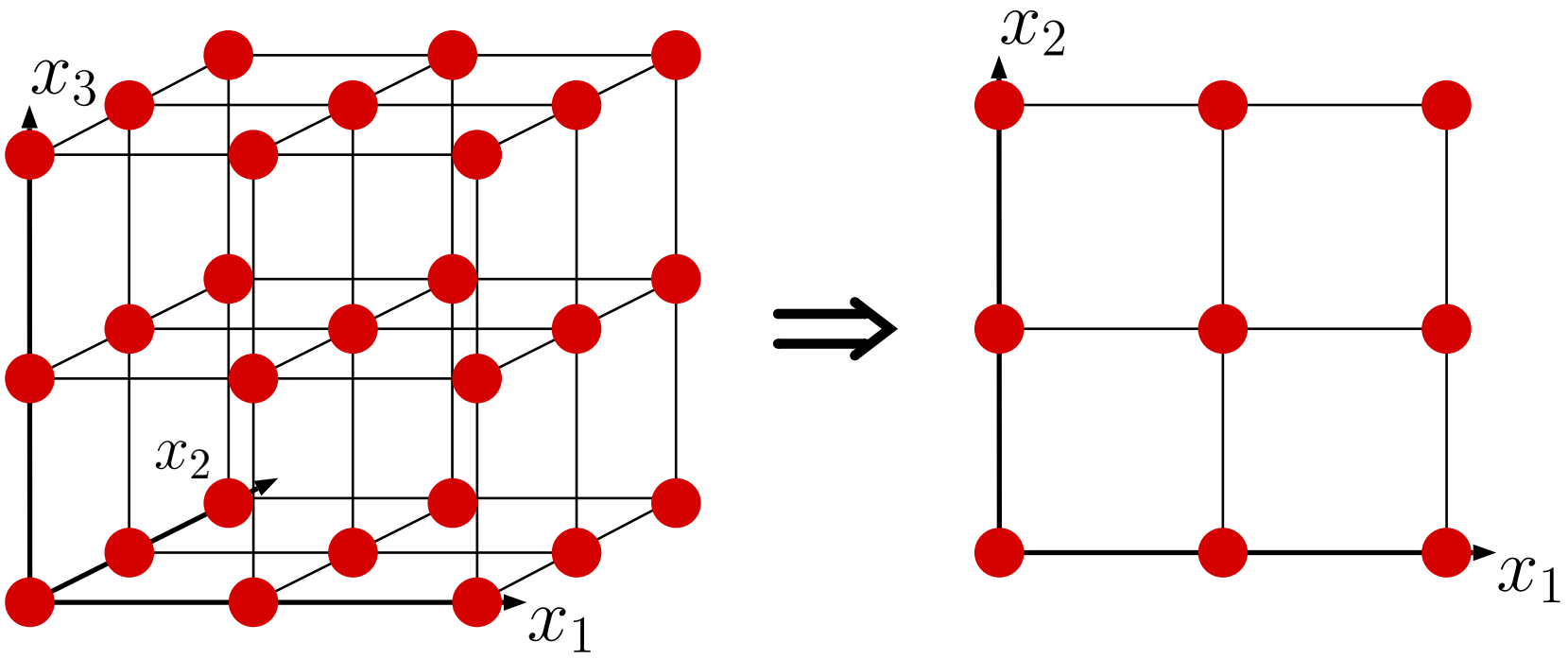
- Easy to use
- adapted to continuous or discrete variables
- Can be combined (star + factorial for example)
- Well suited (often optimal) for linear regression

cons:

- Number of evaluation is not flexible
- Number of evaluation too large in high dimension
- Points are on top of each other when projected

projection issues

Why don't we want points to be superimposed when projected?
If one of the variables has no influence, most observations become redundant...



From 27 observations, we end up with only 9...

Space filling DoE

We will now focus on designs of experiments that:

- are not model oriented
- give information about every domain of the input space
- have good projection on subspaces
- have a flexible number of points

How can we evaluate if a set of points fills the space?

Option 1. Compute distances between points

maximin the minimum distance between two points of the design should be large:

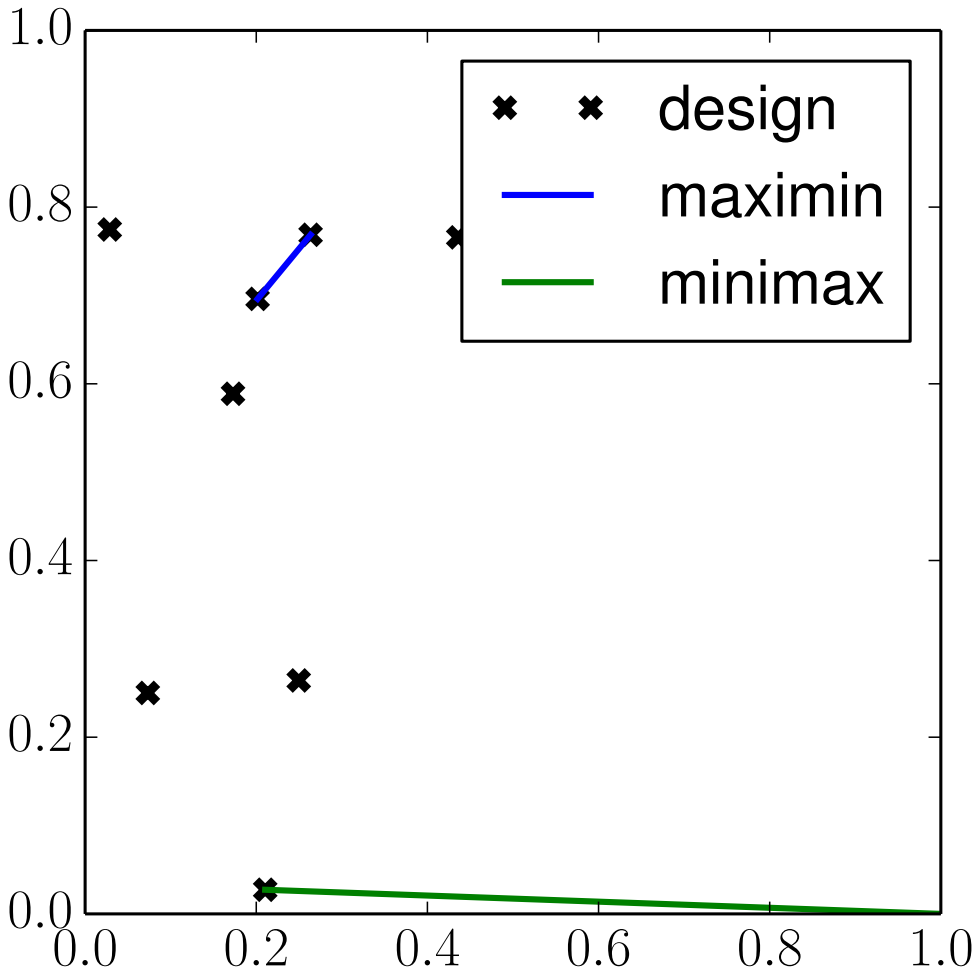
$$\text{Optimisation problem is:} \quad \max_{X_1,\dots,X_n} [\min_{i \neq j} dist(X_i, X_j)]$$

minimax the maximum distance between any point of the space and closest design point should be small:

$$\text{Optimisation problem is:} \quad \min_{X_1,\dots,X_n} (\max_{x \in D} [\min_{i} dist(x, X_i)])$$

The second criterion is much more difficult to optimise

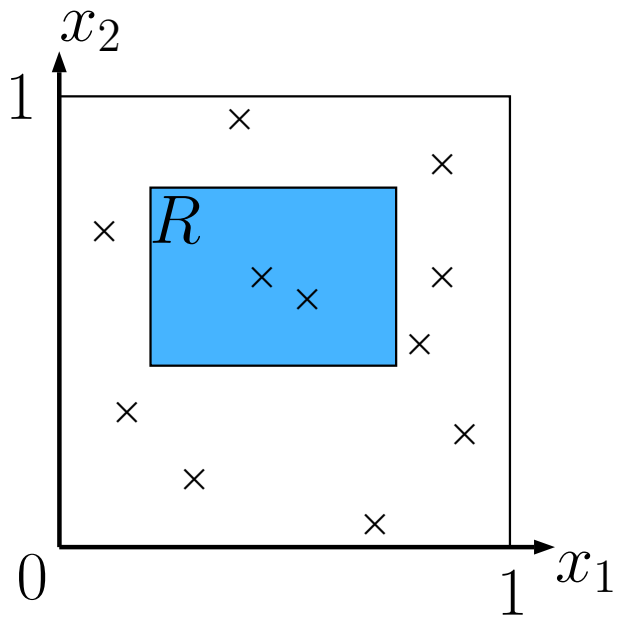
These criteria can be illustrated on a simple 2-dimensional example



How can we evaluate if a set of points fills the space?

Option 2. Compare the distribution with an uniform distribution

Discrepency is a measure of non uniformity. It compares the number of points in a hyper-rectangle with the expected number of samples from a uniform distribution



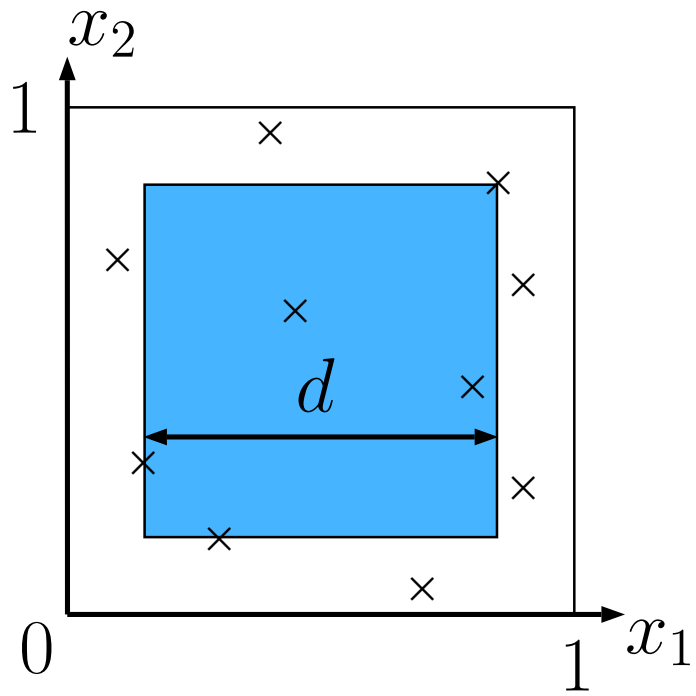
The probability for a uniform variable to be in R is 0.22 and we observe an empirical probability of 2/11. The discrepancy (w.r.t. R) is then:

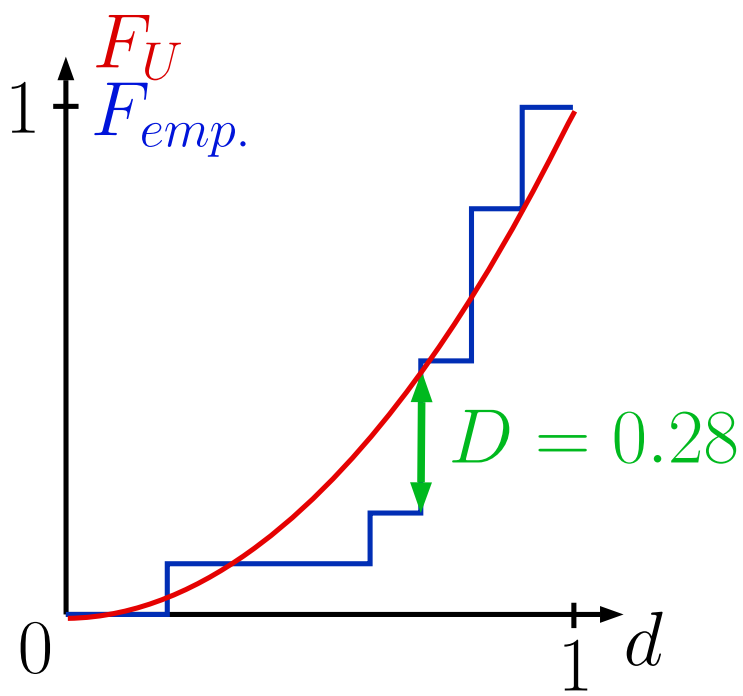
$$D_R = |0.22 - 2/11| = 0.038$$

Discrepency is defined as the sup of the distance between the empirical and analytical cdf.

Discrepency is often computed by:

- fixing one of the hyper-rectangle summit at the origin
- centering the hyper-rectangle





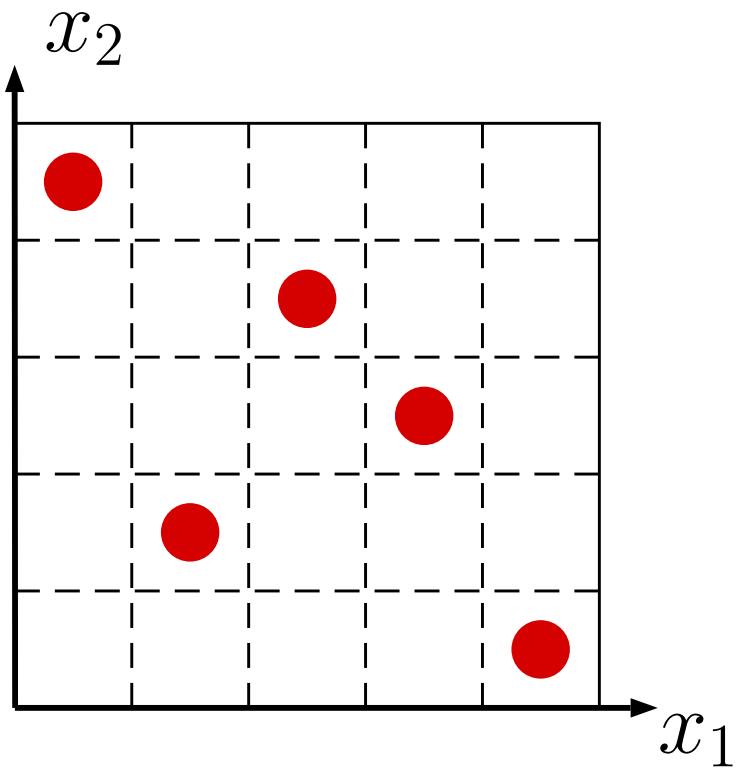
The maximum is located where the rectangle is tangent to points
$\rightarrow$ The optimisation is over a finite space

We will now give an overview on three types of space filling designs:

- Latin hypercubes
- low discrepancy sequences
- centroidal Voronoi tesselations

Latin hypercubes

Latin hypercubes are designs where the domain is sliced in n^d blocks and where there is only one point per "line" and "column":



These designs have good projection properties

A well known example of LHS in 2D is... Sudoku

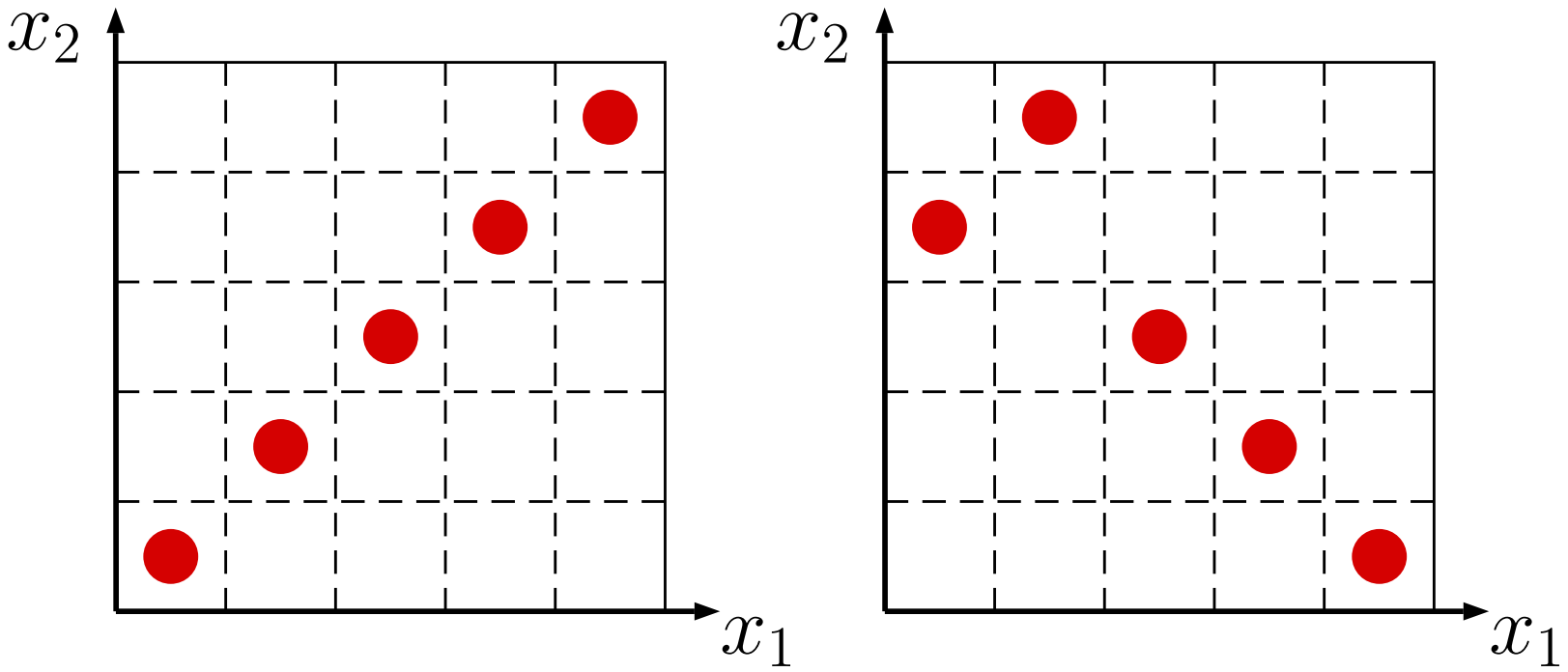
4	3	1	6	7	9	5	2	8
9	6	7	2	5	8	3	4	1
5	8	2	1	4	3	9	6	7
6	5	9	8	1	7	2	3	4
3	2	8	5	6	4	1	7	9
7	1	4	9	3	2	8	5	6
8	7	3	4	2	1	6	9	5
1	4	5	3	9	6	7	8	2
2	9	6	7	8	5	4	1	3

If we focus on one digit (say 4), we obtain a LHD:

●	3	1	6	7	9	5	2	8
9	6	7	2	5	8	3	●	1
5	8	2	1	●	3	9	6	7
6	5	9	8	1	7	2	3	●
3	2	8	5	6	●	1	7	9
7	1	●	9	3	2	8	5	6
8	7	3	●	2	1	6	9	5
1	●	5	3	9	6	7	8	2
2	9	6	7	8	5	●	1	3

Sudoku have more properties that LHD: the generalisation is called **orthogonal array**.

Latin hypercubes do not necessarily cover the space very well...



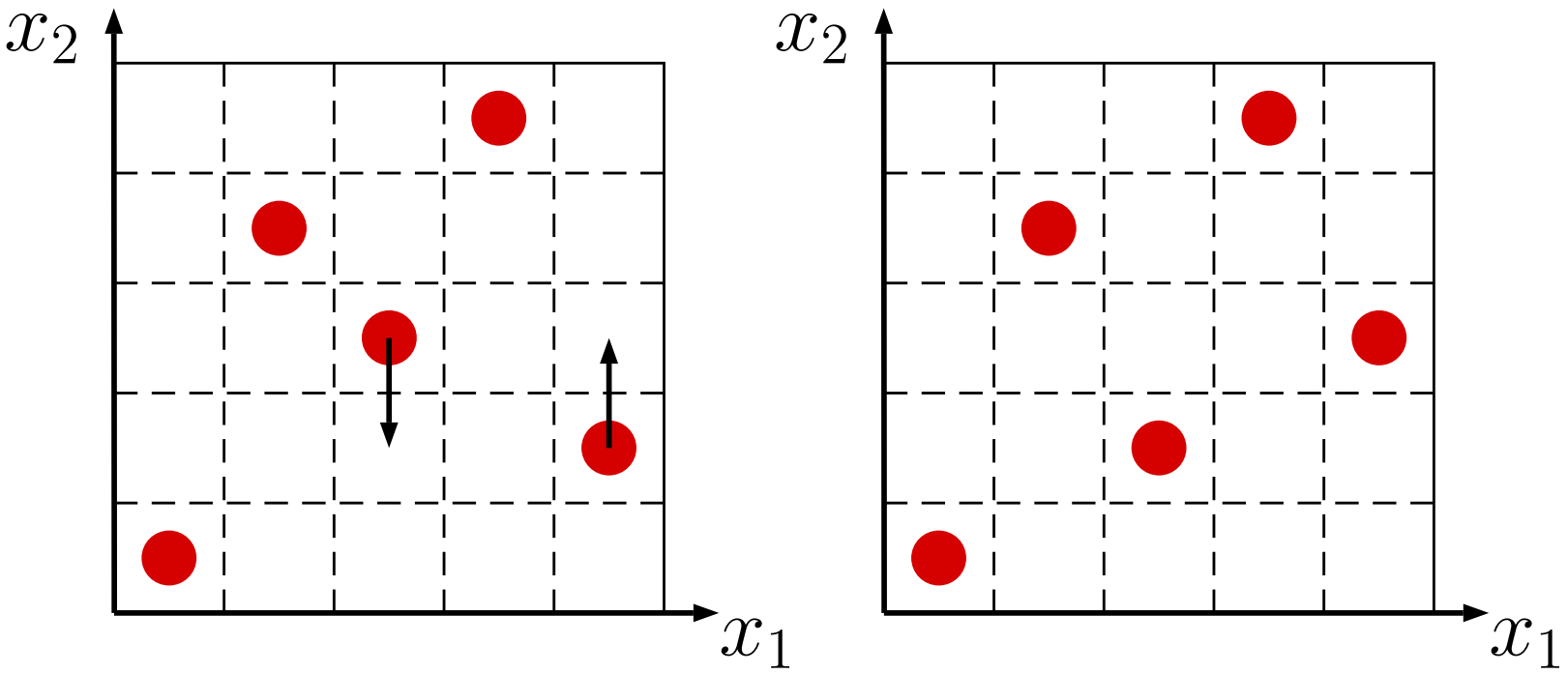
They have to be combined with a criterion such as maximin.

Exercise

- Generate a 5 points LHD in dimension 3.
- How would you program a function `LHD(n,d)`?
- How would you optimize a LHD to ensure it fills the space properly?

Solution

The coordinates of two points can be exchanged:



LHD optimization with simulated annealing:

Morris and Mitchell Algorithm

1. Generate LHD
2. find "bad" points according to maximin
3. choose randomly a column of this critical point and exchange it with an randomly selected other point
4. if the criteria is improved, the modification is accepted
5. otherwise, it is accepted with a probability of

$$\exp\left(\frac{maximin_{new} - maximin_{old}}{T}\right)$$

Low discrepancy sequences

Low discrepancy sequences are deterministic sequences that converge toward the uniform distribution.

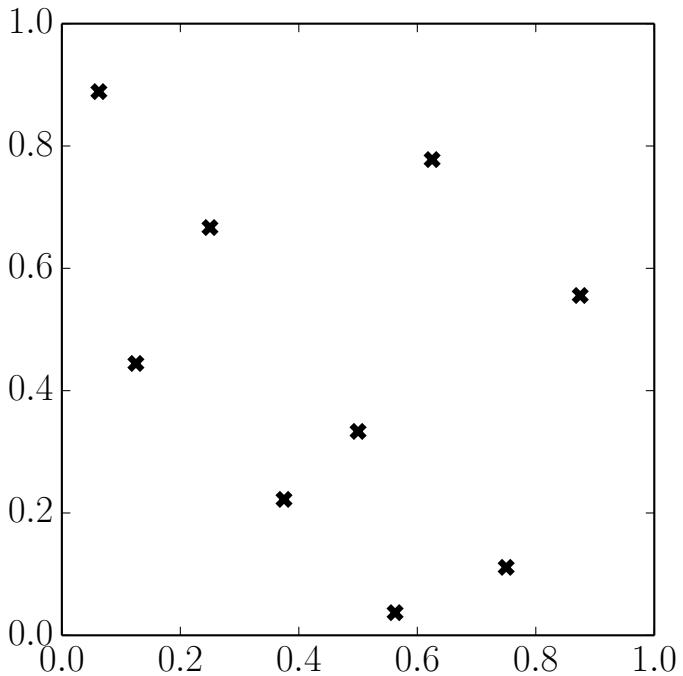
- They cover the space quickly and evenly
- They are easy to build
- It is easy to add new points

Many low discrepancy sequences can be found in the literature: Halton, Hammerley, Sobol', Faure, van der Corput, ...

Example (Halton sequence)

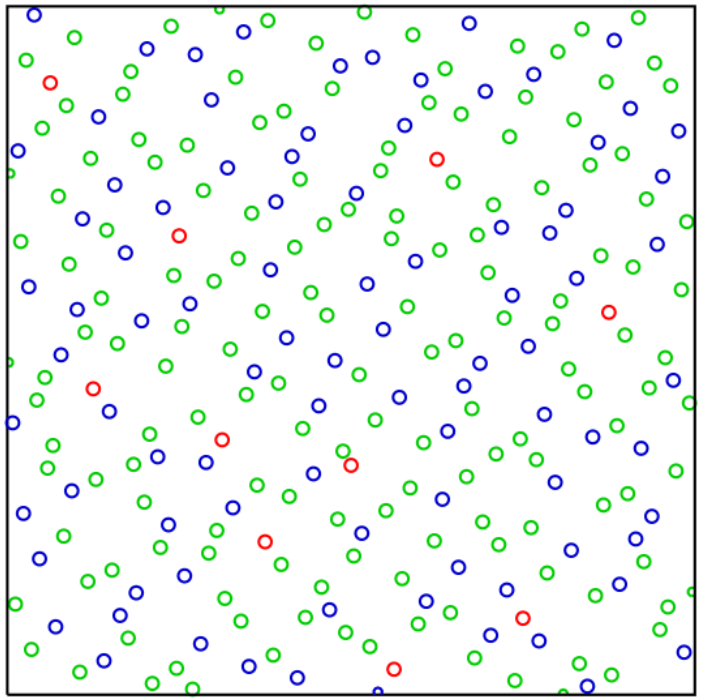
Let a and b be two integers with no common dividers (say 2 and 3). The x_1 and x_2 coordinates of the Halton sequence are:

$$x_1 = 1/2,\ 1/4,\ 3/4,\ 1/8,\ 5/8,\ 3/8,\ 7/8,\ 1/16,\ 9/16, \ldots$$
$$x_2 = 1/3,\ 2/3,\ 1/9,\ 4/9,\ 7/9,\ 2/9,\ 5/9,\ 8/9,\ 1/27, \ldots$$

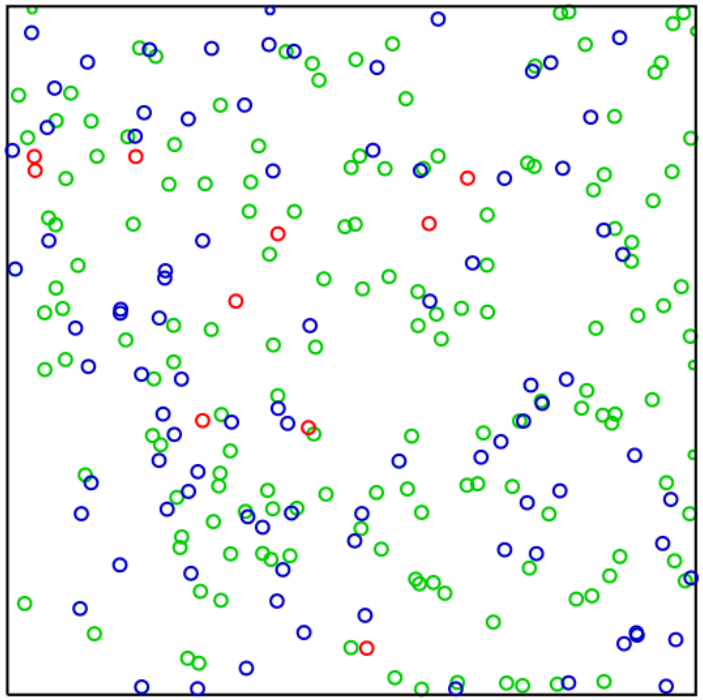


Example (Halton sequence)

Halton Sequence



uniform pseudo random



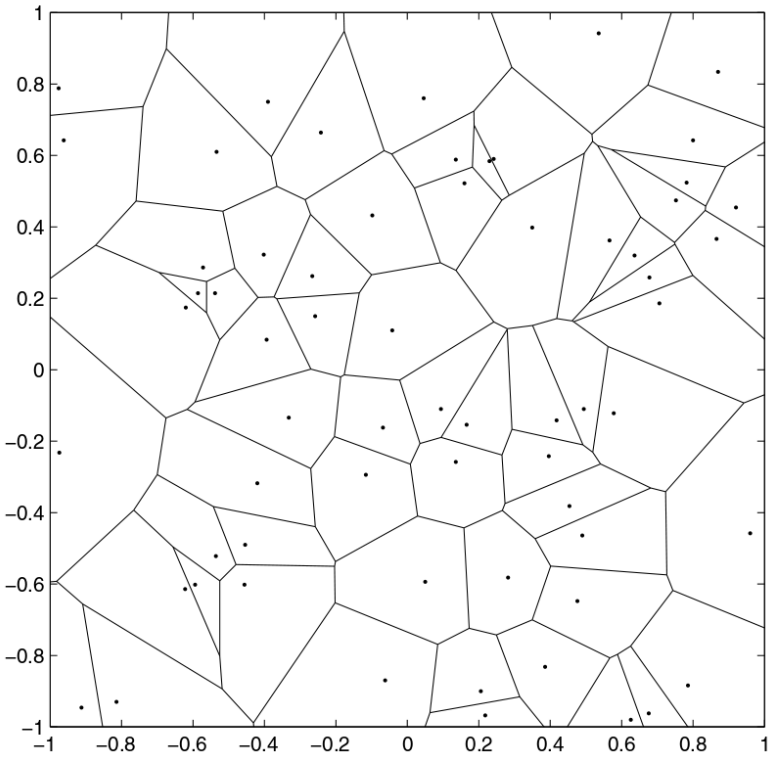
source: wikipedia

Issues with low discrepancy sequences:

- there can be alignments when projected
- there can be holes in subspaces
- points may be aligned (Example: 16 first points in basis (17,18))

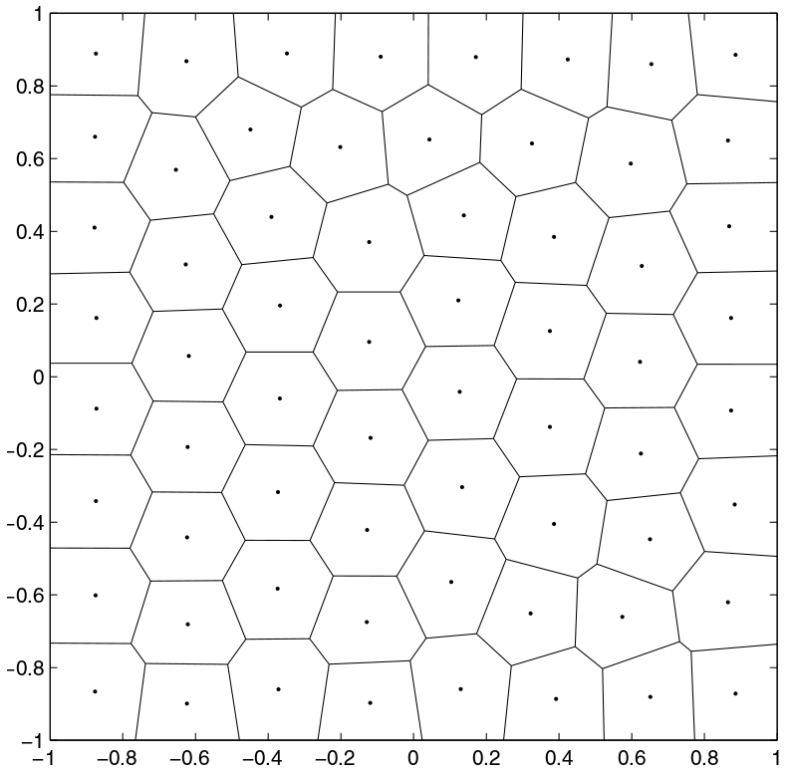
Centroidal Voronoi Tesselations

Given a set of generative points X, the **Voronoi Tesselations** (or Voronoi cells) associated to the point X_i is the region of the space such that X_i is the closest point from the set:



Source: Q. Du et Al., *Centroidal Voronoi Tessellations: Applications and Algorithms*, SIAM Review, 41-4, 1999.

Centroidal Voronoi Tesselations (CVT) is a special case of Voronoi Tesselations where the generative points correspond to the centre of mass of the cells



Source: Q. Du et Al., *Centroidal Voronoi Tessellations: Applications and Algorithms*, SIAM Review, 41-4, 1999.

Properties of CVT:

- Each point of the space is close to one generative points
- The generative points cover the space

$\Rightarrow$ The generative points of CVT can be used as design of experiment.

Generating CVT

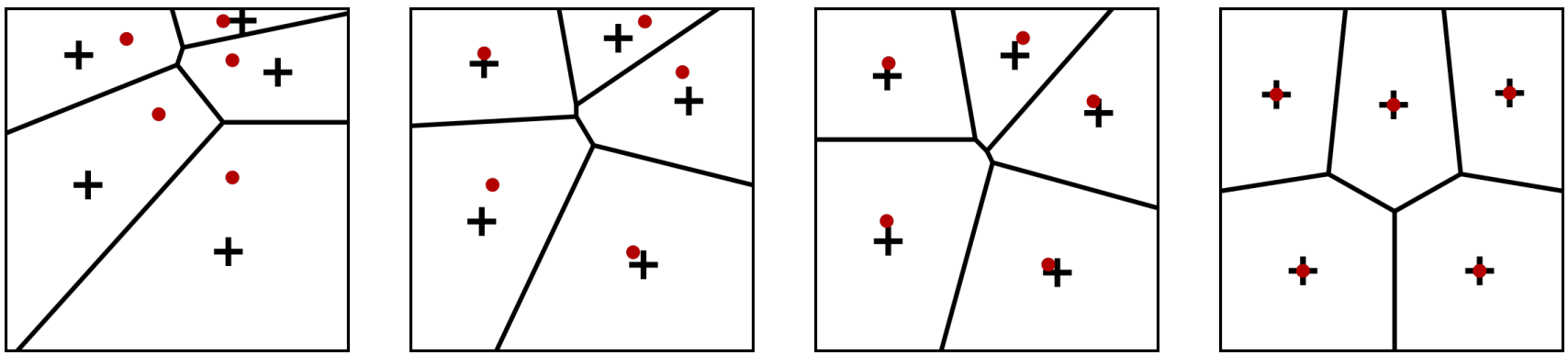
1. Lloyd's Algorithm

1 Initialize X as a set of n points

2 While $i < nb\_iter$

3 Compute the Voronoi diagram of X

4 X = centre of mass of each cell



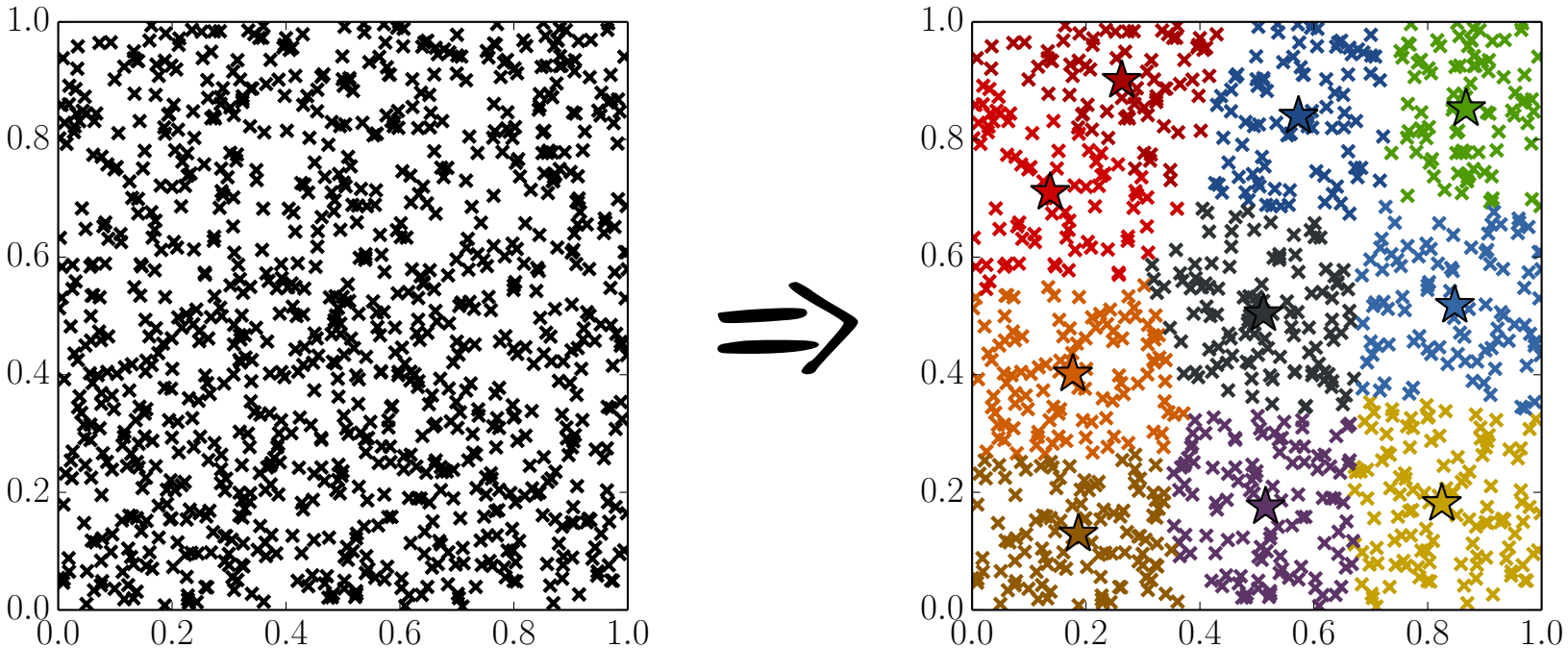
iteration 1 iteration 2 iteration 3 iteration 15

source: "Lloyd's algorithm" wikipedia page

Generating CVT

2. k-means

This algorithm is very similar to Lloyd but it uses a large set of points covering the input space instead of the full continuous domain:



Generating CVT

3. McQueen algorithm

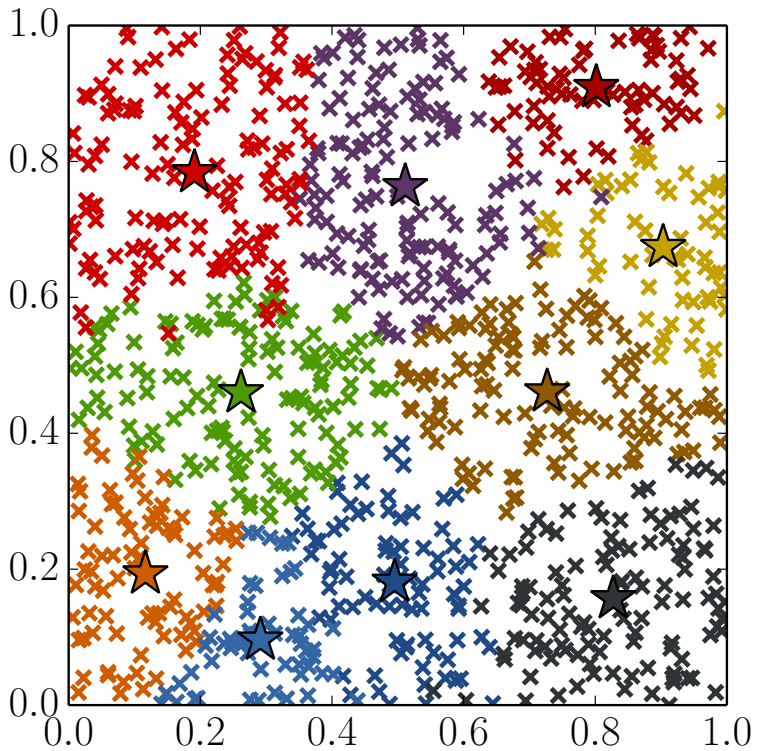
This algorithm is much faster than the previous ones and gives a good approximation

1. Initialize X as a set of n points
2. Initialize k as a vector of 1 with length n
3. While $i < nb\_iter$
4. generate one random point z in the input space
5. find the X_i closest to z
6. update $X_i = \frac{k_i X_i + z}{k_i + 1}$
7. $k_i = k_i + 1$

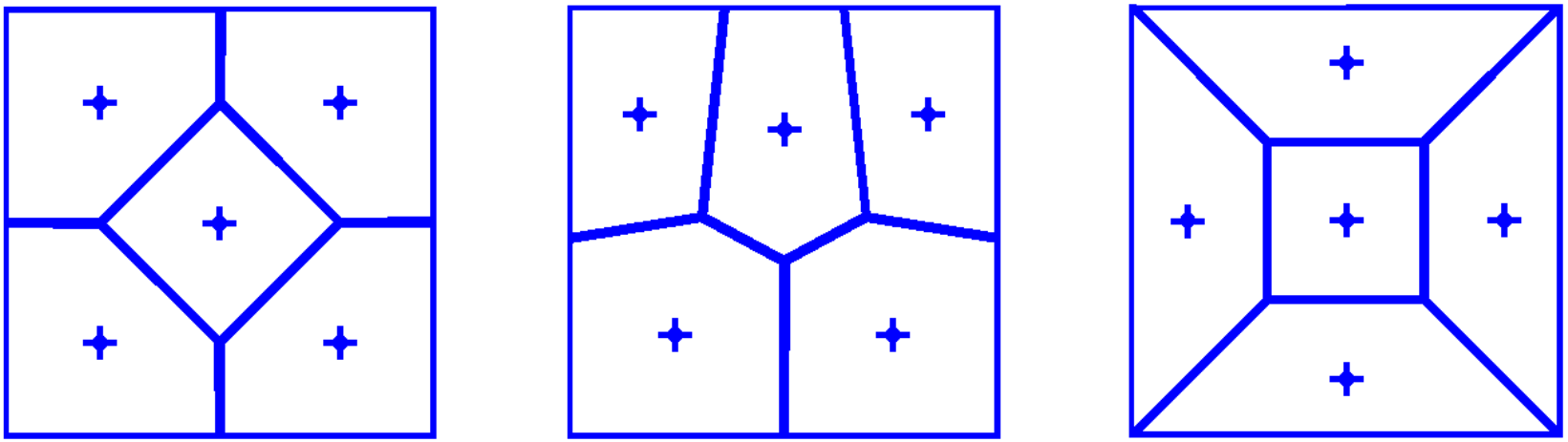
Generating CVT

3. McQueen algorithm

We obtain the following design:



CVT are not unique:



source: wikipedia page “Centroidal Voronoi Tesselations”