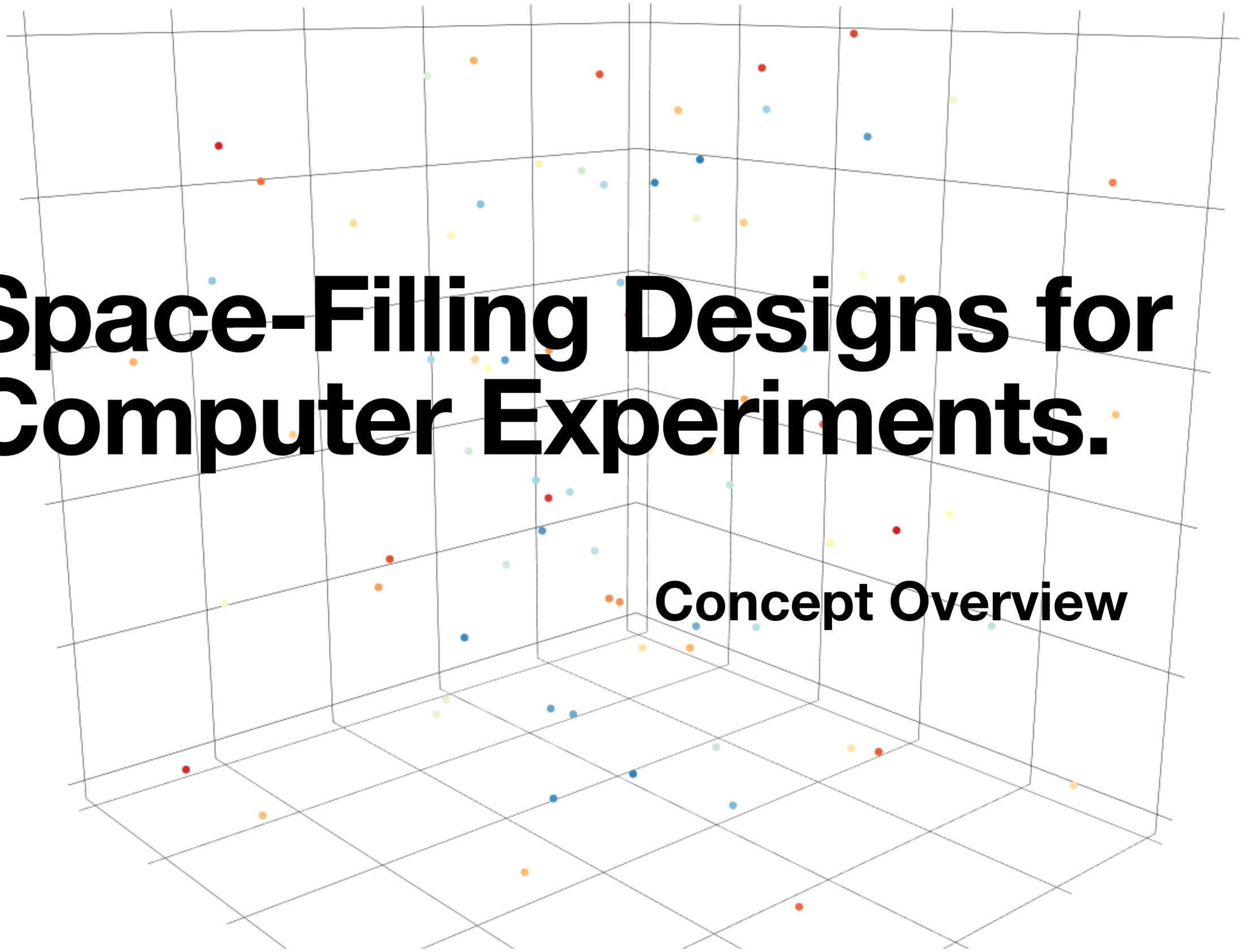

Space-Filling Designs for Computer Experiments.

Concept Overview

Goal : Spread simulation runs evenly across the input space so the surrogate model can learn the full response surface.

- Running a computer experiment can be expensive
- We cannot afford many simulation runs (in our case we are limited to 200 runs)
- We need just enough points to build a reliable surrogate model (Kriging)

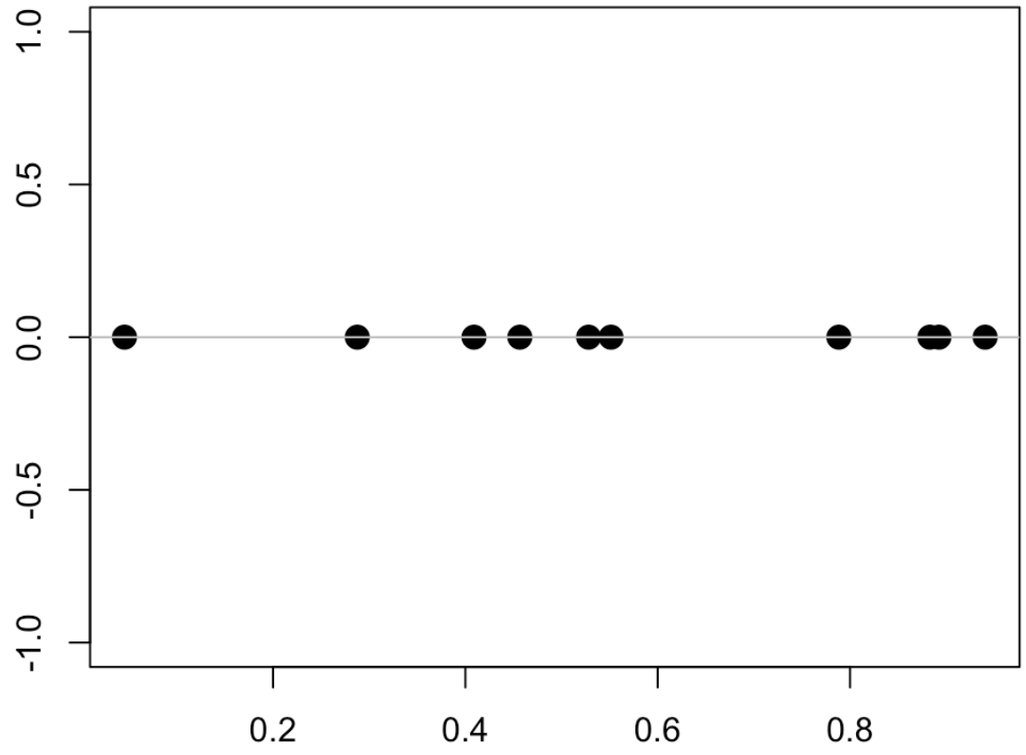

Fig 1: 10 Random points in 1D

So what sample size should we use?

Rule of Thumb for Sample Size:

Use ≈ 10 x dim points

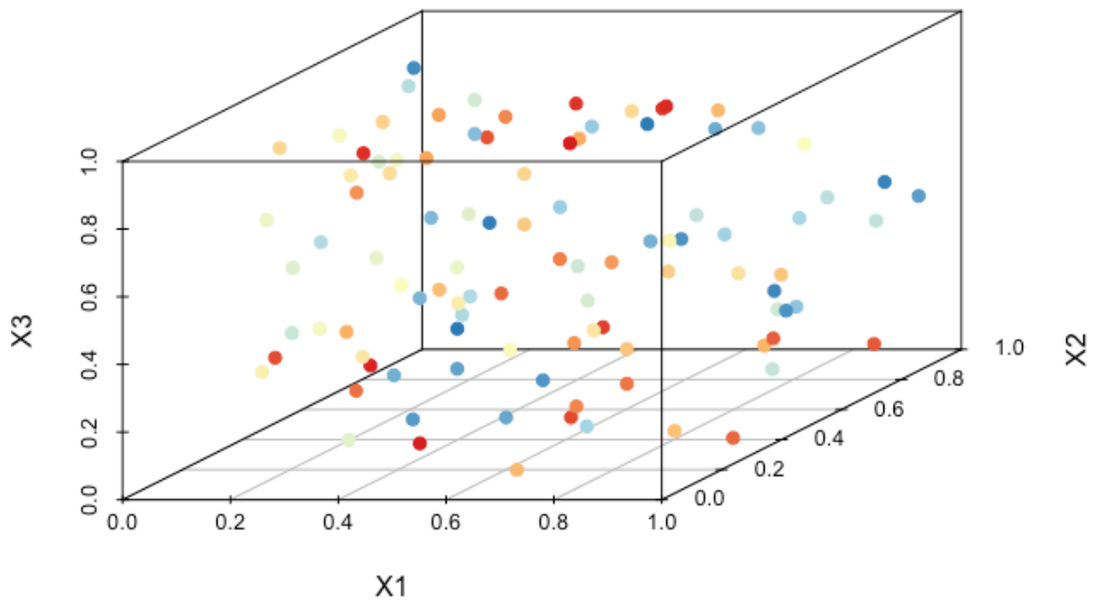

Fig 2: 100 Random point in 3D

Space-Filling Designs: Making Your Simulation Points Stop Acting Like Introverts at a Party

Why this matters? Because…

- Surrogate models (Gaussian Process, RBF, etc.) need data from all regions.
- If points cluster, the model knows only a small part of the input space.
- If there are holes, prediction error becomes large in those regions.
- Space-filling designs reduce uncertainty, bias, and prediction error.

Okay, but how do we come up with Space-Filling Designs?

1. We start random.
    - Simple Random Sampling
    - Stratifies Random Sampling
2. Latin Hypercube Sampling: Like Random Sampling, But With Rules.
3. Distance Criteria: Making Your Points Stop Sitting on Each Other.
    - Maximin (Maximise the minimum distance)
    - Minimax (Minimise the maximum hole)

And many more…

Latin Hypercube Sampling

- Divide each input variable's range into n_s equal intervals
- Select exactly one sample point from each interval in every dimension
- Ensures perfect 1D coverage for each input variable
- Pros: Even coverage along each input axis, Good starting point for building space-filling designs
- Cons: Not fully space-filling by itself, needs some kind of optimisation.

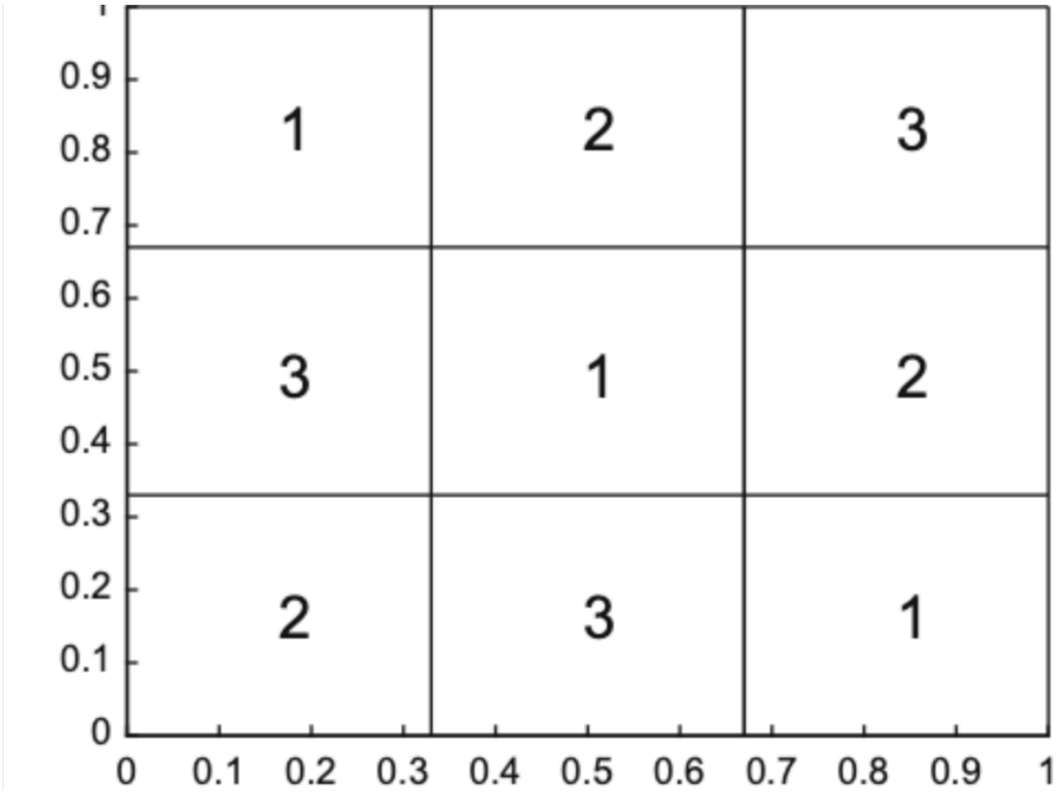

Fig 3: Latin Hypercube

Distance criteria:

Maximin - Maximise the minimum distance between any two design points

- Pros: Very effective for exploration, good for finding patterns. Cons: Optimises only the single smallest distance

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

Fig 4: P-th order distance between x_1, x_2 ∈ X

- **Minimax** - Minimise the maximum distance from any in the input space to the nearest design point

- Pros: Excellent for surrogate modelling, design fills in underserved regions. Cons: Computationally expensive

Fig 5: ChatGPT Explained It… So Here We Are.

Maximin spreads points apart. Minimax fills the holes. Together they keep your design from being a disaster.