

# Constrained Design of Experiments for Data-Driven Models

Fabian Schneider<sup>1,3</sup>, Max Schüssler<sup>1</sup>, Ralph Hellmig<sup>2,3</sup>,  
Oliver Nelles<sup>1</sup>

<sup>1</sup>Universität Siegen, Department Maschinenbau,  
Institut für Mechanik und Regelungstechnik – Mechatronik  
Paul-Bonatz-Str. 9-11, 57068, Siegen, Germany

E-Mail: {fabian2.schneider,max.schuessler,oliver.nelles}@uni-siegen.de

<sup>2</sup>Universität Siegen, Department Maschinenbau,  
Lehrstuhl für Materialkunde und Werkstoffprüfung  
Paul-Bonatz-Str. 9-11, 57068, Siegen, Germany  
E-Mail: ralph.hellmig@uni-siegen.de

<sup>3</sup>EJOT SE & Co. KG  
Im Herrengarten 1, 57319, Bad Berleburg, Germany  
E-Mail: {fschneider,rhellmig}@ejot.com

## Abstract

The quality of data-driven models depends significantly on the data distribution in the input space. In this work, design of experiments (DoE) methods for constrained input spaces are discussed. An approach based on an Latin hypercube (LH) design is introduced to deal with strongly constrained input spaces. For the unconstrained case, where the input space is a hypercube, different design of experiments methods have been developed. The dominating state-of-the-art methods are Sobol sequences and Latin hypercubes. Instead of optimizing complete LH designs, the proposed strategy is to incrementally construct an LH design. Every new sample is selected by a distance-based metric. The presented method is then applied in two test cases and compared to a method based on a Sobol sequence. Here, an initial design is created by a

DOI: 10.58895/ksp/1000151141-14 erschienen in:

**Proceedings – 32. Workshop Computational Intelligence: Berlin, 1. - 2. Dezember 2022**

DOI: 10.58895/ksp/1000151141 | <https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000151141/>

Sobol sequence and every sample is removed that violates the constraints. The design qualities are measured by the resulting model accuracies of data-driven models. A function generator is applied to create synthetic data sets to train and evaluate local linear model networks.

## 1 Introduction

Models are a basic component of several advanced techniques in many disciplines. They are utilized in applications like simulation, optimization, control, and fault detection tasks. The requirements for model accuracy and complexity increase in all these applications. As a result, deriving a model by first order principles is often not applicable, because the derivation of such models is often too complicated or the model evaluation is too slow. Data-driven models which rely only on measurements are more and more frequently the better choice. If a model relies solely on measurement data, the importance of good and efficient measurement plans is obvious. In the scope of this work, a measurement can be a real physical measurement or created by computer experiments. For example by the evaluation of first principle models like the finite element analysis. A measurement plan is also called experimental design. To create such a design, different design of experiment strategies have been developed [9].

A standard design space which is also the input space of the model is a  $d$ -dimensional hypercube. If no prior knowledge for a process exists, uniformly distributed experimental designs should be used [9]. Sobol sequences are a quite popular choice and an example for a low-discrepancy series. These designs are uniformly distributed over the input space, even for high-dimensional problems. Optimized Latin hypercube (LH) designs achieve also good results in numerous applications. The maximin-criterion is commonly used for optimization [12]. It maximizes the minimal distance between two samples of the design. One drawback of those maximin-LHs is the need for solving an optimization problem. It is hard to solve and also computationally expensive. The difficulties increase with the dimensionality and design size. The perfect one-dimensional projection property of LHs is one advantage for modeling tasks. Those two mentioned methods are suited for unconstrained input spaces.

Constraints leads to an unregular  $d$ -dimensional input space. As a result new or adapted methods must be applied for such tasks.

The constraints lead also to a conflict of objectives. It is not possible to create an uniformly distributed design over the feasible space with uniformly distributed one-dimensional projection properties. Two different methods are compared in this work to analyze this conflict. The first method is based on a Sobol sequence. After an initial design is created, every sample violating the constraint is removed from the design. This is an easy method. And this should lead to an approximately uniformly distributed design over the feasible regions of the input space. However, depending on the constraints, the projections are not uniformly distributed. The second method is a version of an LH design. In an incremental procedure, new samples are added to the design. Out of a set of feasible new samples, in every iteration the sample is selected by the  $\phi_p$ -criterion [15]. Therefore, the sample is selected that has the largest distance to its nearest neighbors.

A function generator is applied, to create test functions. These are used to build synthetic data sets. The created designs are evaluated by a comparison of the test errors of local linear models for those generated functions.

## 2 Design of Experiments

Design of experiments covers all methods and strategies to create an experimental plan. In the early 20th century, Fisher [4] introduced a method to investigate statistical properties of crop systems. In recent time, DoE strategies are also applied to plan computer experiments with the goal building a meta or surrogate model. Computer experiments are in most cases deterministic. Computer experiments, like Finite Element Analysis, are often computationally very expensive. So, one target of DoE for computer experiments is to create a design that extracts the necessary information to model the process with as few samples as possible. Passiv learning strategies are the standard case. The design is defined before the training of the model has started or the data have been measured or computed. All regions of the input space are weighted equally. Thus, an uniformly distributed design is desired [9]. Besides passiv learning strategies, active learning strategies have been developed [1]. In

contrast to passiv learning strategies, active learning strategies are incremental. The selection of every new sample or a new batch of samples is for example based on the model quality. Utilizing the model error as criterion increases the sample density in regions of high model errors. The process output could also serve as a criterion. In model based optimization tasks, regions of the input space with undesired process outputs do not need to be modeled with the same accuracy as regions of high interest. Designs created by passive methods typically serve as initialization of active strategies.

Different evaluation metrics of the design distribution have been developed for unconstrained design spaces. Common are discrepancy based methods or metrics, like the centered L2 discrepancy [15] which describes the uniformity of the design over the input space and of all projections over the corresponding subspaces. The Kullback-Leibler-Divergence is also applied frequently. It is a distribution based method. The design is asset by comparing the actual sample distribution with the desired one. As a third group, distance-based metrics exists. Here, the distances between the samples of the design serves as basis for evaluation. The  $\phi_p$ -criterion belongs to this group [15]. Constraints often prevents the application of those standard methods. The estimation of sample distributions over a constrained input space is in complex cases not feasible. Kernel density estimation methods are not suited for unregular input spaces. Boundary effects impair an accurate computation [10].

### $\phi_p$ -criterion

It is not desirable that samples of a design are close to each other, because then, it is expectable that only few new information is generated by each sample. Thus, generally a design with large nearest neighbor distances (NN) for all samples is preferred. A common strategy to achieve large NN distances is to maximize the minimal NN distance. Unfortunately, if a sample is modified to improve the distance to its actual nearest neighbor, it is possible that another sample becomes the new NN. In this case, the NN distance criterion is non-smooth. Therefore, all gradient-based solution strategies are inapplicable. Additionally, only the minimal NN distance is considered by this procedure. All remaining samples are irrelevant.

Therefore, the  $\phi_p$ -criterion has been developed [6] to obtain a smooth approximation of the minimal NN distance criterion:

$$\phi_p(\mathbb{U}) = \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^N \|\underline{u}_j - \underline{u}_i\|^{-p} \right\}^{1/p}, \quad (1)$$

where  $\mathbb{U}$  is the set of samples of the design and each sample is defined as  $\underline{u}_i = [u_{i,1} \ u_{i,2} \ \cdots \ u_{i,d}]^T$ . While the minimal NN distance has to be maximized, the  $\phi_p$  criterion has to be minimized. The price to be paid is a higher computational load because all sample pairs have to be evaluated in (1) and no efficient K-D Tree search algorithms can be utilized. This formulation is better suited for optimization tasks. The weighting of the nearby samples increases with growing values for  $p$ . Commonly used values are  $p = 15 \dots 50$  [6]. In this work, all studies have been carried out with  $p = 15$ .

## 2.1 Sobol Sequence

Sobol proposed this sequence [11] in 1967. It is a low-discrepancy, quasi-random design and is commonly applied for modeling tasks. Numerical integration tasks are a further common application. Compared to modeling tasks, the dimensionality of integration problems is often larger. As a result, Sobol provided sequences for up to 51 dimensions. This sequence is designed to create samples as uniformly distributed as possible over the input space [11].

## 2.2 Latin Hypercube Sampling

Latin hypercube designs are constructed to ensure perfect one-dimensional projection properties. LHs are per definition non-collapsing designs. To create an  $d$ -dimensional Latin hypercube with  $N$  samples, each dimension is divided into  $N$  levels. Each sample of the design is, as originally proposed, created by a random combination of the available levels in each dimension.

By this procedure, designs may occur that have poor space filling qualities [12]. To avoid such designs, optimization strategies have been developed. The resulting optimization problems are hard to solve. An overview over

different methods is given in [12]. In [3], an LH design is optimized by coordinate swapping. The method is called Extended Deterministic Local Search algorithm (EDLS). The optimization target is to create a maximin design by optimizing the  $\phi_p$ -criterion, see (1), of the design  $\mathbb{U}$ . The EDLS method outperforms many other LH designs in the unconstrained case. Especially, the translational propagation (TPLHD) algorithm [13] and the iterated local search (ILS) algorithm [5].

In the unconstrained case a coordinate swapping is always possible. In the constrained case, both modified samples have to satisfy all constraints after the swapping. If the input space is highly restricted, most swapping operations are infeasible which leads to significantly poorer design qualities.

## 2.3 Incremental Latin Hypercube

The incremental Latin hypercube (ILH) algorithm is proposed in the following. In [14], an incremental methodology is presented to build LHs. Here, the dimensionality of the design and the design size grow in each increment.

The in this work presented method constructs a design in an incremental way, instead of modifying a complete LH design. In every iteration, a new sample is added to the design. As a result, no coordinate swapping is necessary. No global optimization is performed.

Similar to the standard LH designs, the interval of each dimension is divided into  $N$  levels. In every iteration, one sample is added to the design. The levels corresponding to this sample are afterwards removed from the set of available levels. As described in algorithm 1, in each iteration a set of possible new samples based on the remaining grid levels is randomly created. The sample of that set is selected that achieves the best  $\phi_p$  value.

For an example of the algorithm see Fig. 1. The target design size is  $N = 5$ . In every iteration  $m = 3$  candidate samples are tested. The constraint, the unfeasible input space, is marked by gray color. The samples of the design at each increment are represented by the crosses. The free levels are marked by dashed gray lines, the already used levels by gray lines, and the levels of the selected point by dotted black lines. The five increments are:

- a) Randomly chosen initial point.

---

**Algorithm 1.:** Incremental Latin Hypercube (ILH)

---

**Input:**  $N$ : design size,  $d$ : dimensionality,  $\mathbb{C}$ : set of constraints,  
 $m$ : number of candidate samples in each iteration

**Output:**  $\mathbb{U}$ : generated design

- 1: Create for every dimension  $k$  a set  $\mathbb{G}_k$  with  $N$  levels (default: equally spaced) in  $[0, 1]$
  - 2: Initialize the design  $\mathbb{U}$  with a random sample  $\underline{u}_1$  by selecting one level of each set  $\mathbb{G}_k$ . This sample has to fulfill the constraints in  $\mathbb{C}$ . Until this sample fulfills the constraint, step 2 is repeated.
  - 3: Remove the chosen levels from each set  $\mathbb{G}_k$ .
  - 4: **for**  $i = 2$  to  $N$  **do**
  - 5:     Build set  $\mathbb{S}$  containing  $m$  candidate samples by choosing random levels in  $\mathbb{G}_k$ .
  - 6:     Remove all samples of  $\mathbb{S}$  that violates any constraint in  $\mathbb{C}$ .
  - 7:     **if**  $\mathbb{S} \neq \emptyset$  **then**
  - 8:         For every element  $\underline{s}_j$  of  $\mathbb{S}$ , calculate  $\phi_{p,j}$  of the design  $\mathbb{U} \cup \underline{s}_j$ , for  $j = 1, \dots, \#\mathbb{S}$ .
  - 9:          $\underline{\tilde{s}}$ : Sample of  $\mathbb{S}$  which achieves best  $\phi_p$  value,  $\min_j \phi_{p,j}$
  - 10:         $\mathbb{U} := \mathbb{U} \cup \underline{\tilde{s}}$ .
  - 11:     Remove each level of  $\underline{\tilde{s}}$  from the corresponding set  $\mathbb{G}_k$ , for  $k = 1, \dots, d$ .
- 

- b) First iteration, triangle, circle, and square are candidate samples. Square lies in the unfeasible region. Triangle achieves lowest  $\phi_p$  value, see Tab. 1. It is added to  $\mathbb{U}$ .
- c) Second iteration, triangle is selected by  $\phi_p$  value and added to  $\mathbb{U}$ .
- d) Only 2 levels are left in each set of levels  $\mathbb{G}_k$ . Therefore, one level is selected 2 times ( $m = 3$ ). Only the triangle fulfill the constraint. It is selected.
- e) For each dimension, only one free level is remaining. Thus, only one candidate sample is created. It violates the constraint. Thus, no sample is added to the design  $\mathbb{U}$ .

Finally, instead of the desired  $N = 5$  samples, the design contains only 4 samples. In this case, it is possible to place all desired 5 samples in the feasible input space. For example, it is possible to place all points on the diagonal line. Please note, that this isn't the normal case. But as illustrated by this example, it is obvious that depending on the selections at earlier increments

Table 1:  $\phi_p$ -values of the candidate samples

Iteration	Triangle	Circle	Square
a)	—	—	—
b)	2	4	unfeasible
c)	4	4	unfeasible
d)	4	unfeasible	unfeasible
e)	unfeasible	—	—

the resulting design size may vary. A determination of the exact design size priorly isn't possible. This algorithm requires only one hyper parameter. It is the parameter  $m$ . It have to be defined depending on the dimensionality and the desired design size. While larger values improve the design quality, they also lead to a rise of the computational costs, because more constraint and  $\phi_p$  evaluations are necessary.

### 3 Function Generator

The evaluation and comparison of different methods in the constrained case is challenging. The comparison in this work is based on test functions. In [2] a function generator is proposed. It is based on polynomial functions and applicable for different dimensionalities. The complexity of the functions can be controlled by the user. The input space of the function is the  $d$ -dimensional hypercube. The functions are generated according to

$$g(u_1, u_2, \dots, u_d) = \sum_{i=1}^M c_i \cdot (u_1 - s_{i,1})^{p_{i,1}} \cdot (u_2 - s_{i,2})^{p_{i,2}} \cdot \dots \cdot (u_d - s_{i,d})^{p_{i,d}}. \quad (2)$$

All parameters are drawn randomly from probability distributions. The coefficients  $c_i$  are drawn from a normal distribution  $N(0, 1)$ , the shifts  $s_{i,j}$  from the uniform distribution  $U(0, 1)$  and the powers  $p_{i,j}$  from an exponential distribution with expected value  $\mu$  and are rounded down [2].  $M$  defines the number of monomials or terms. This and  $\mu$  controls the complexity of the generated



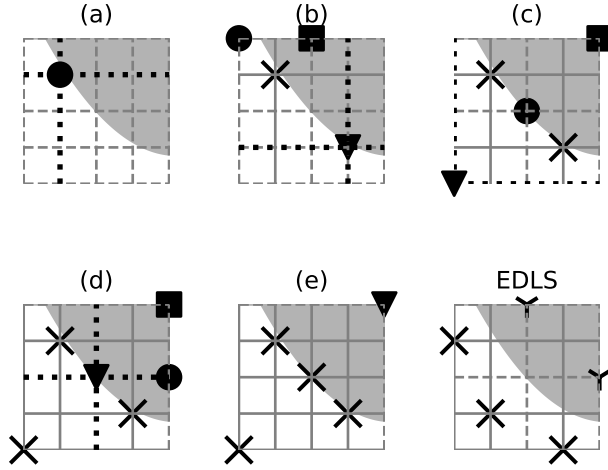


Figure 1: A  $5 \times 5$  example of the ILH algorithm. Plots (a) to (e) describe the five increments. The last plot shows an via EDLS optimized LH. Here, tow samples are unfeasible.

functions. In the scope of this work, all functions are created with  $M = 10$  and  $\mu = 2$ . This parameter combination results in complex functions. Some examples are shown in Fig. 2. The function outputs are normalized to the interval  $[0, 1]$ .

## 4 Results

First, the unconstrained case is used to study some properties of the different methods. With these insights the ILH is applied to 2 different test constraints. The first one is a two-dimensional problem. The input space is constrained by a polynomial function. The resulting shape is comparable to an efficiency map of a combustion engine with the two inputs rotation speed and torque. The second application is a parametrization of a geometrical body. The body is parameterized by five parameters. Thus the input space is 5-dimensional. The volume of the generated solids has to be constant. Only small tolerances are allowed. So, not all parameter combinations are feasible. Such a constraint or

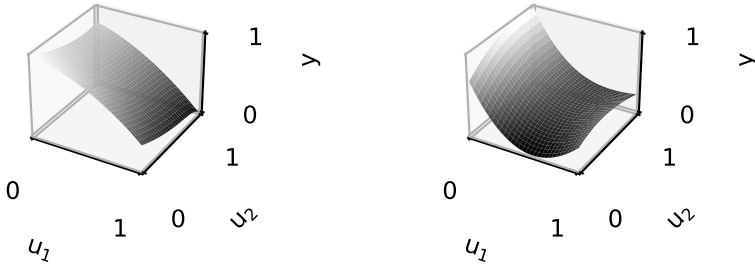


Figure 2: Two two-dimensional normalized functions generated with  $M = 10$  and  $\mu = 2$ .

similar ones could occur for example in forming processes. Here, the volume constancy of the processes restricts the valid shapes of the solids.

#### 4.1 Studies of the Unconstrained Case

To understand the constrained case easier, some properties of the different design methods and the  $\phi_p$ -criterion, see (1), are analyzed.

$\phi_p$  is often selected as a criterion in LH optimizations. It drives the samples as far away from each other as possible. This behavior is shown by the following example. Based on a Sobol sequence  $\mathbb{U}_{sob}$ , a test design  $\mathbb{U}_\phi$  is created. The first data sample of  $\mathbb{U}_{sob}$  initializes  $\mathbb{U}_\phi$ . In an incremental procedure, beginning from the second sample of  $\mathbb{U}_{sob}$ , every sample  $\underline{u}_i$  is optimized. The objective is to minimize  $\phi_p$  of the merged design  $\mathbb{U}_\phi \cup \underline{u}_i$  according to

$$\begin{aligned} & \min_{\underline{u}_i} \phi_p(\mathbb{U}_\phi \cup \underline{u}_i) \\ & \text{subject to} \\ & 0 \leq u_{i,k} \leq 1, k = 1 \dots d. \end{aligned}$$

The solution of the optimization problem, the resulting sample, is added to  $\mathbb{U}_\phi$ . Afterwards these steps are repeated with all remaining samples. Each sample of  $\mathbb{U}_\phi$  is placed, such that it minimizes  $\phi_p$  at the current iteration. This is similar to a maximization of the distance to the nearest neighbor of  $\underline{u}_i$ . The one- and

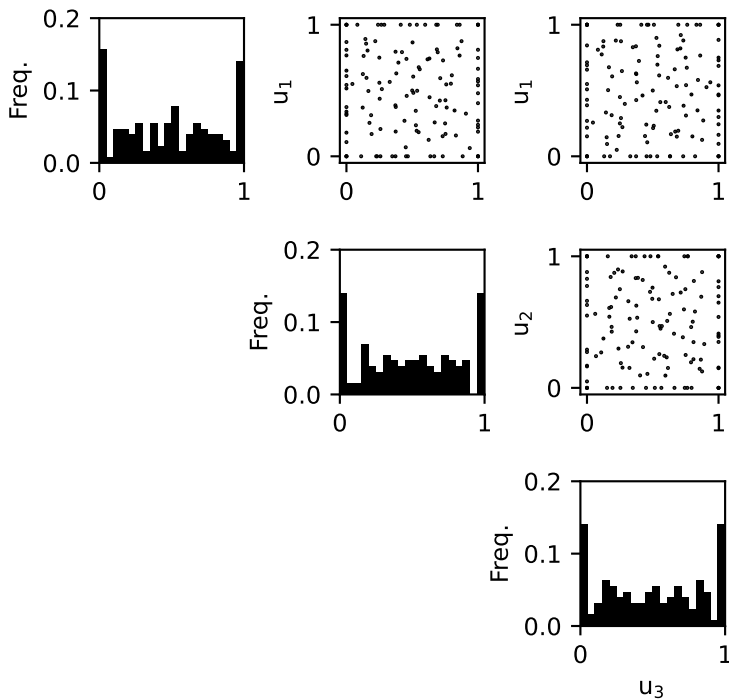


Figure 3: Projections of a three-dimensional design with 128 samples incrementally optimized with regard to the  $\phi_p$ -criterion.

two-dimensional projections of such a three-dimensional design are shown in Fig. 3. Obviously, the optimization drives the samples near to the boundaries of the hypercube. The one-dimensional projections are not uniformly distributed at all. According to [2] a design with samples in the edges and, corners of the input space is only preferable if large extrapolation is demanded. The curse of dimensionality implies that with increasing dimensionality a larger volume of the input space is near to the boundaries of the hypercube. Thus, the behavior already observed in the three-dimensional case for the one-dimensional projections would further increase with the dimension. The LH design enforces per definition a uniform distribution of the one-dimensional projections. This

could be interpreted as a regularization of the  $\phi_p$ -criterion. Accumulations of samples close to the boundaries are prevented.

Based on this observations, it is expectable that the nearest neighbor (NN) distance of each sample grows with the distance to the center of the hypercube. In Fig. 4 the nearest neighbor distance of each sample is shown for different distances to the center of the input space. The distances are normalized by the maximal possible distance, the distance from the center of the hypercube to a corner. A Sobol sequence, a random LH design, the presented ILH design and a via the EDLS algorithm [3] optimized LH are compared. The results of the Sobol sequence and the random LH are as expected. The nearest neighbor distance grows with increasing distance to the center. Also the variance in each group is comparably large. The EDLS design instead surprises. The variance of the nearest neighbor distance is smaller. This is a result of the optimization. But, not expected, the nearest neighbor distances over all intervals are nearly constant and overall larger compared to the other methods. This might be another reason besides slightly better one-dimensional projection properties why EDLS optimized LHs are preferable for modeling tasks compared to the Sobol sequences [3]. This property might be even more important for kernel-based models like Gaussian process regression models. The ILH design achieves also better results as the Sobol sequence overall. Compared to the EDLS design, the NN distances are slightly smaller and the variance is larger.

## 4.2 Parabola Constraint

In this section, the ILH design is applied to a single constraint in 2D. It is motivated from typical maps of combustion engines with maximum power constraints. All samples  $i$  in the unit hypercube that also fulfill

$$-(1.3 \cdot (u_{i,1} - 1)^2 + 0.2 - u_{i,2}) \leq 0 \quad (3)$$

are feasible design points.

In Fig. 5 a Sobol sequence (all unfeasible samples are removed), and two ILH designs are presented. An EDLS design is not applicable because the constraint prohibits most of the swapping options. This makes an optimization unfeasible. For both strategies, the resulting number of samples is not known

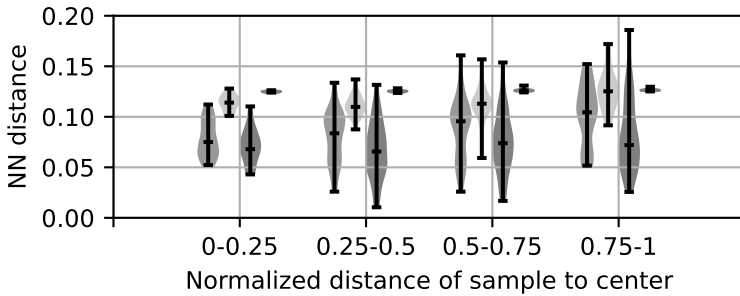


Figure 4: Nearest neighbor distances of every sample over the distance of the sample to the center. Each design is three-dimensional and consists of 512 samples. From left to right: Sobol sequence, ILH, random LH, EDLS optimized LH

exactly before the construction of the designs. The sizes of the initial designs are chosen such, that the resulting design has a size of 95 % to 100 % of the target size.

To compare the quality of the created designs, data-driven models are trained. The function generator is applied to create 20 different artificial processes. A local linear model network with an incremental tree construction algorithm (HILOMOT = Hierarchical Local Model Tree) [7] is trained for every process. In this study, 10 different ILH designs are evaluated. In Fig. 5 two out of those 10 ILH designs and the design based on the Sobol sequence are shown. Designs with sizes from 64 to 2048 are created. The test errors of the trained models are presented in Fig. 6. In total, the ILH design achieved better results than the Sobol sequence in 58% of the 1200 different ILH designs. For two exemplary models, the absolute values of model errors are shown in Fig. 7. The largest error occurs in the lower left corner of the model trained with the Sobol sequence. In accordance with the results of the comparison of the nearest neighbor distances it is expected that more samples of the ILH designs are close to the boundaries of the input space. In these regions, the model quality with the ILH is generally higher.

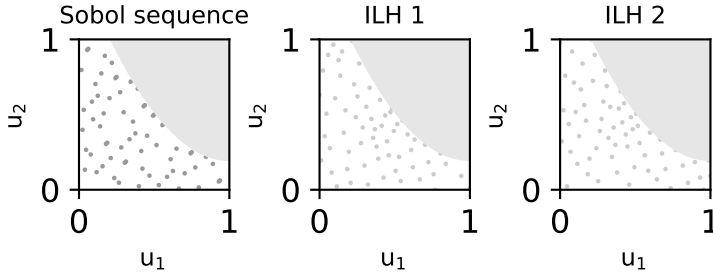


Figure 5: Different designs created, each contains 128 samples. Sobol sequence (left one) and two different ILH designs.

### 4.3 Volume Constancy

In the second test case, a 5-dimensional and constrained input space is analyzed. The five inputs describe the shape of a rotationally symmetric solid. Basis splines (B-splines) are applied to describe the contour of the part, see Fig. 8. For more information on B-splines see for example [8]. A parametrization is necessary to perform a geometry optimization. In the field of forming or forging processes the volume of the created parts is often restricted. It has to be constant over the forming or forging process. This requirement restricts the allowed parameter combinations significantly. In the test case, the volume tolerance is  $\pm 2\%$ . In Fig. 9 two-dimensional projections of the constrained design space are illustrated. The solid is manipulated by the position of the marked control points (CP) of the B-splines. The assignment with the allowed value ranges of the CPs to the design is given in Tab. 2. The designs for this application are evaluated similarly to the previous. 10 different ILH designs and one Sobol sequence are constructed. 20 different test functions with  $M = 10$  and  $\mu = 2$  are generated. An overview of the results is given in Fig. 10. In this application, the ILH designs outperform the Sobol sequence in 78% of the cases. The one-dimensional projections of both methods differ significantly, see Fig. 11. The projections of the ILH design are more uniform distributed than the projections of the Sobol sequence. The nearest neighbor distances

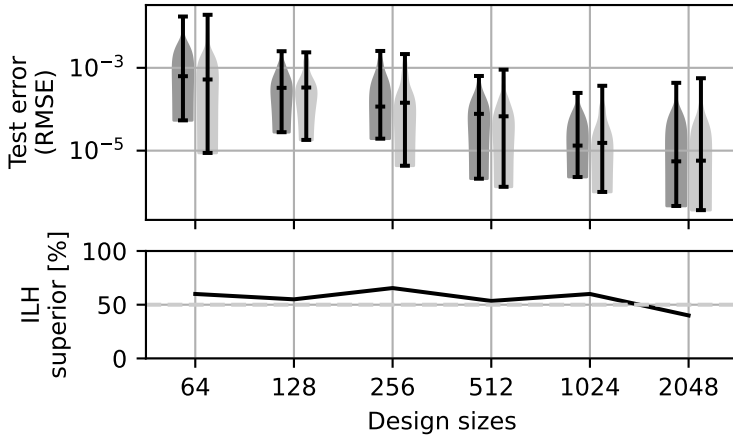


Figure 6: Model errors of all trained models. Dark gray and left: test errors of models trained with Sobol sequence. Light gray and right: test errors of models trained with ILH designs. In the lower plot, the probability is given that a model trained with an ILH design is superior to the model trained with the Sobol sequence.

are also larger for the ILH designs. These results match the observations of the unconstrained case. They suggest that the one-dimensional projection properties has a significant influence on the model quality.

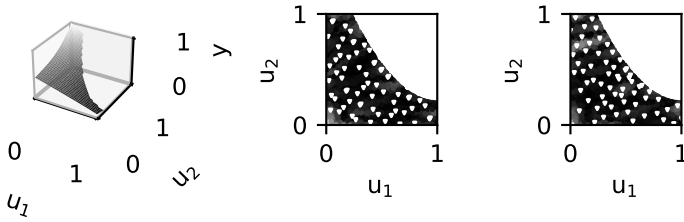


Figure 7: Left plot: test process generated by function generator. Centered plot: Head map of absolute model errors of a model trained with a design based on a Sobol sequence. Right plot: Head map of absolute model errors of a model trained with an ILH design. The samples of each design are visualized by the white triangles.

Table 2: Parametrization of the geometric solid

Control point		Coordinate	Min	Max
$u_1$	0	y	3.7	4.7
$u_2$	1	x	3.9	5
$u_3$	2	x	4	6
$u_4$	1	y	1.5	4
$u_5$	2	y	0.5	2.4

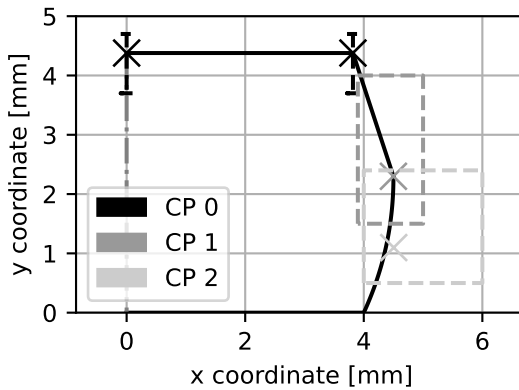


Figure 8: Parametrization of the test case. Parameterized control points with allowed coordinate ranges are shown.



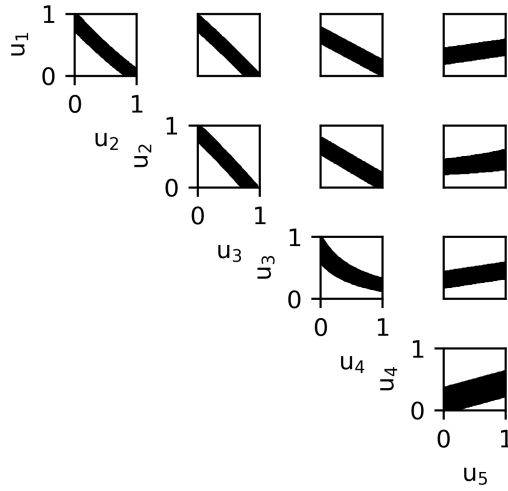


Figure 9: Two-dimensional projections of the input space based on the center point of the input space. Black areas represents the feasible region.

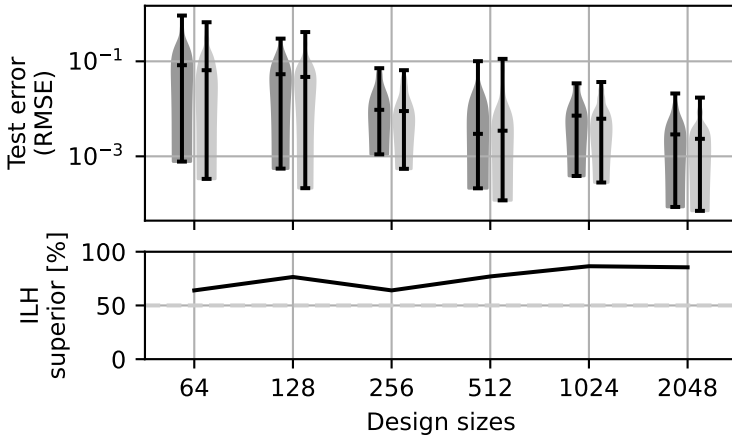


Figure 10: Model errors of all trained models. Dark gray and left: test errors of models trained with Sobol sequence. Light gray and right: test errors of model trained with ILH designs. In the lower plot, the probability is given that a model trained with an ILH design is superior to the model trained with the Sobol sequence.

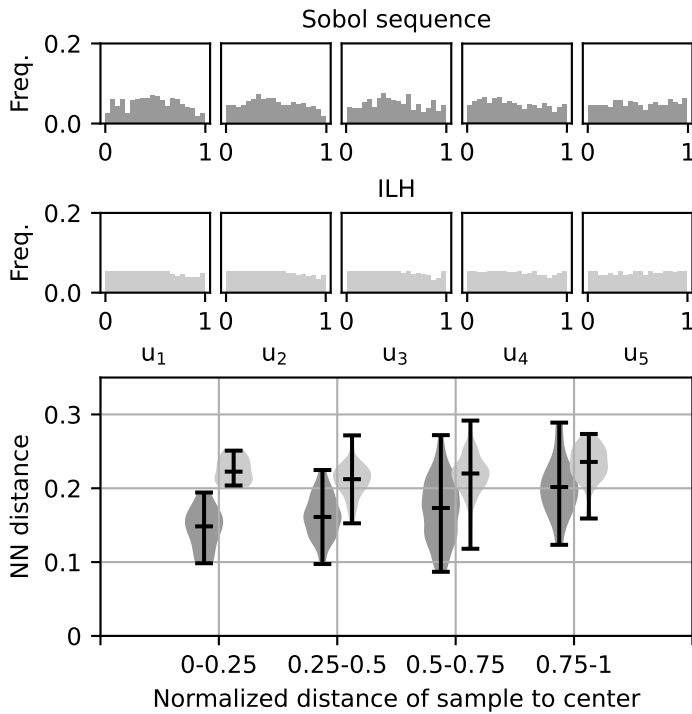


Figure 11: One-dimensional projections and nearest neighbor distances (Sobol sequence left, ILH right) of the Sobol sequence and a randomly selected ILH design with 512 samples.

## 5 Conclusion

Two algorithms for design of experiments for constrained input spaces are compared in this work. They have been applied to two test processes, a two-dimensional and a 5-dimensional. The quality of the created designs has been evaluated on the basis of the test error of one data-driven models. Due to the limitations of different evaluation criteria for constrained design of experiments, a function generator is applied to create test processes for the evaluation of different designs. A novel strategy to create Latin hypercubes for constrained design spaces has been proposed. The algorithm is based on an incremental construction strategy. In each increment, the best sample is added to the design. This is selected by the  $\phi_p$ -criterion from a randomly generated set of candidate samples. These proposed incremental Latin hypercube (ILH) designs achieve lower model errors for both applications. The one-dimensional projections of the ILH designs are more uniformly distributed than the projections of the design based on Sobol sequences. The nearest neighbor distances of the ILH designs are also higher compared to Sobol sequences.

## References

- [1] J. Belz. *Fighting the curse of dimensionality with local model networks*. PhD thesis, Universität Siegen, 2018.
- [2] J. Belz and O. Nelles. Proposal for a function generator and extrapolation analysis. In *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6, 2015.
- [3] T. Ebert, T. Fischer, J. Belz, T. O. Heinz, G. Kampmann, and O. Nelles. Extended deterministic local search algorithm for maximin latin hypercube designs. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 375–382, 2015.
- [4] R. A. Fisher. The arrangement of field experiments. 1992.

- [5] A. Grosso, A. Jamali, and M. Locatelli. Finding maximin latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, 197(2):541–547, 2009.
- [6] M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.
- [7] O. Nelles. Axes-oblique partitioning strategies for local model networks. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 2378–2383, 2006.
- [8] L. Piegl and W. Tiller. *The NURBS Book*. Springer Berlin Heidelberg, 1995.
- [9] L. Pronzato and W. G. Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, apr 2011.
- [10] D. W. Scott. *Multivariate Density Estimation*. Wiley, aug 1992.
- [11] I. Sobol’. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.
- [12] F. Viana. Things you wanted to know about the latin hypercube design and were afraid to ask. 05 2013.
- [13] F. A. C. Viana, G. Venter, and V. Balabanov. An algorithm for fast optimal latin hypercube design of experiments. *International Journal for Numerical Methods in Engineering*, 82(2):135–156, 2010.
- [14] T. Voigt, M. Kohlhase, and O. Nelles. Incremental latin hypercube additive design for lolimot. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1602–1609. IEEE, 2020.
- [15] M. Vořechovský and J. Eliáš. Modification of the maximin and  $\phi_p$  (phi) criteria to achieve statistically uniform distribution of sampling points. *Technometrics*, 62(3):371–386, 2020.