Proc. International Joint Conference on Neural Networks (IJCNN), Vancouver, Canada, 2016

Order of Experimentation for Metamodeling Tasks

Julian Belz, Konrad Bamberger, Oliver Nelles
Department of Mechanical Engineering
University of Siegen
D-57068 Siegen, Germany
E-mail: julian.belz@uni-siegen.de

Order of Experimentation for Metamodeling Tasks

Julian Belz, Konrad Bamberger, Oliver Nelles
Department of Mechanical Engineering
University of Siegen
D-57068 Siegen, Germany
E-mail: julian.belz@uni-siegen.de

***Abstract*—The order in which measurements are carried out, determines the accuracy of models in early stages of the measurement process, i.e. while measurements are still in progress. Reliable models in early stages of the data acquisition phase allow for model-based investigations like optimization runs or an earlier switching to an active learning phase. This paper compares different methods to determine the order of experimentation for regression problems in metamodeling tasks. The data distribution and the data density in the input space are varied for several randomly generated synthetic functions in order to find the most promising determination strategy for the order of experimentation. As an application example, all strategies are also applied to a computational fluid dynamics (CFD) metamodel. The order of experimentation based on the intelligent k-means clustering algorithm turns out to be the best overall order-determination strategy.**

I. Introduction

In the context of experimental modeling (identification), gathered data plays the key role to build reliable models. Therefore an experimental design has to be determined in order to set locations in the input space, where information should be collected. An often neglected aspect and main topic of this paper regards the order in which the measurements are conducted to yield the best possible model performance with fractions of the whole experimental design. No literature could be found addressed to order of experimentation aiming at this specific goal. For example, [1] and [2] consider the order of experimentation only in the context of neutralizing influences of undesirable factors on the experimentation or efforts needed to change factor levels. Additionally, these two publications deal only with factorial designs, whereas methods proposed in this paper can be applied to arbitrary experimental designs. The method presented in [3] aims at a reduction of the training set size in order to decrease computational demands and to improve convergence speed of the model training. In differentiation to that, our proposed method aims to improve the convergence with respect to the *data amount* and does not consider computational demands at all. Good models in early stages of the measurement process yield several advantages, e.g. time can be saved because demanded model qualities can be reached with less data. In addition the model can be used earlier, while the measurement process is still in progress. For example, model-based optimization runs become more reliable with small data subsets. Active learning strategies enlarge the experimental design in an iterative and adaptive way, based on models trained with the currently available training data, such that the information gained by additional measurements is maximized. Typically, an initial experimental design is needed before the active learning phase can start [4]. Through a good order of experimentation the necessary amount of data serving as initial experimental design might be decreased. Note that throughout this paper the term "measurement" is used synonymously for obtained data points, regardless of their origin. For example, the data might originate from real-world measurements as well as from computer simulations.

This paper compares different methods to determine the order of experimentation for regression problems in metamodeling tasks. Metamodels try to describe the true input/output relationship of deterministic computer simulations [5]. Therefore, a metamodel is a computationally inexpensive "model of a model", that tries to approximate a computationally expensive simulation with high accuracy [6]. Since all data gathered for the generation of a metamodel is completely deterministic, basic principles of experimental design for controlling noise and bias like replication, blocking or randomization don't have to be considered [7]. In contrast to real-world test benches, there are no resources required to change the value of any actuating variable for computer experiments. These facts simplify the order of experimentation determination in the special case of metamodeling tasks. However, an extension of the order determination strategies proposed in this paper for real-world measurement scenarios is possible, as will be outlined in Sec. V.

II. Strategies for the Order of Experimentation Determination

The goal of all strategies is to achieve the best possible model performance with subsets of an already determined (and fixed) experimental design. Therefore it is assumed, that it is beneficial to gather information from all areas of the input space as soon as possible to achieve reliable metamodels in early stages of the measurement process. Having a similar task for real-world applications, more restrictions and effects resulting from environmental influences have to be considered for the order of experimentation determination, as already outlined in the introduction.

In the following, all strategies for the order of experimentation determination used for this paper are explained. For the explanations it is always necessary to distinguish between three sets of data points. Set $\mathbb{N}$ contains all points of a data set. $\mathbb{S}$, containing all already sorted points, and

$\mathbb{F}$, containing all not yet sorted points, are non-overlapping subsets of $\mathbb{N}$. Here and throughout the rest of the paper, "sorted" refers to the successional order of experimentation, i.e. the order in which the measurements should be conducted. The relationships between the previously defined sets can mathematically expressed as:

$$\mathbb{N} = \mathbb{S} \cup \mathbb{F} \text{ and } \mathbb{S} \cap \mathbb{F} = \emptyset \ . \tag{1}$$

At the beginning of each method, $\mathbb{F}$ contains all data points. Then points are sequentially moved from $\mathbb{F}$ to $\mathbb{S}$ until $\mathbb{F}$ is empty.

A. Biggest Gap Sequence (BGS)

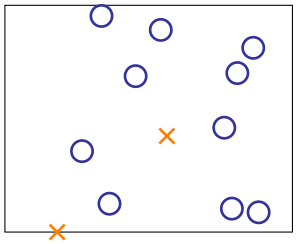
For the biggest gap sequence (BGS), the first point to be added to $\mathbb{S}$ is the one closest to the center of all data points. In the following, one iteration of the BGS is explained and illustrated in Fig. 1. Here one iteration refers to all steps necessary to determine one data point, that should be added to the sorted list next. In Fig. 1a there are two already sorted points in $\mathbb{S}$ (orange crosses), whereas all other points still belong to set $\mathbb{F}$ (blue circles). Each point in $\mathbb{F}$ is now assigned to its nearest neighbor (NN) from subset $\mathbb{S}$. The dashed lines in Fig. 1b connect each point in $\mathbb{F}$ with its NN in $\mathbb{S}$. The corresponding distances (lengths of the dashed lines) from all points in $\mathbb{F}$ to their NN in $\mathbb{S}$ are calculated. The point with the maximum distance to its NN is selected and is moved from $\mathbb{F}$ to $\mathbb{S}$, see Fig. 1c. After adding a point to $\mathbb{S}$, the next iteration starts and the whole procedure continues until $\mathbb{F}$ is empty. Note that $\mathbb{S}$ is incremented in size only after the last step of each iteration.

B. Median Distance Sequence (MDS)

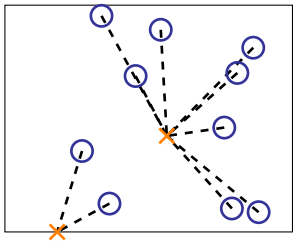
The median distance sequence (MDS) starts similar to the BGS algorithm, i.e. the first point added to $\mathbb{S}$ is the one closest to the center of all data points. After that, each point in $\mathbb{F}$ is assigned to its NN from subset $\mathbb{S}$, like in the BGS strategy, see Fig. 1b. Again, the distances from all points in $\mathbb{F}$ to their NN in $\mathbb{S}$ are calculated. Now the point that corresponds to the median value of all calculated distances is moved from $\mathbb{F}$ to $\mathbb{S}$, instead of the one with the maximum distance. An update of the NN relationships between $\mathbb{F}$ and $\mathbb{S}$ has to be performed. After that the procedure continues until $\mathbb{F}$ is empty.

C. Intelligent k-means Sequence (IKMS)

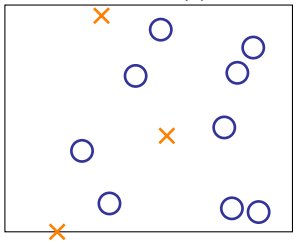
Clusters according to the intelligent k-means algorithm [8] are determined. The number of clusters n_c is automatically chosen by this algorithm. Then all data points belonging to a cluster C_i, $i = 1 \ldots n_c$ are sorted according to the BGS strategy described in Subsection II-A. As a result there are n_c sorted data point lists or cluster lists respectively. To get the final order of experimentation, the first element of each list is added to $\mathbb{S}$. In that way the first n_c points to be simulated are the ones closest to the cluster centers. After that, the second element of each cluster list is added until all data points are ordered according to the intelligent k-means sequence (IKMS). Figure 2 demonstrates the IKMS procedure for 12 points. First



(a) Two already sorted points (x)



(b) Assigning each point in $\mathbb{F}$ (o) to its NN in $\mathbb{S}$ (x)



(c) Maximum NN is added to $\mathbb{S}$ (x)

Fig. 1: Illustration of the procedure for the MDS and BGS strategies

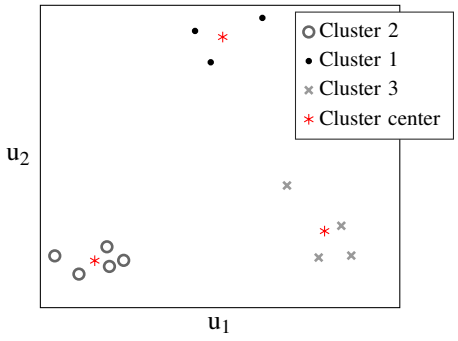
the intelligent k-means algorithm determines three clusters and assigns each point to one cluster, see Fig. 2a. The numbers shown in Fig. 2b represent the final ordering determined by the IKMS algorithm.

D. Random Sequence

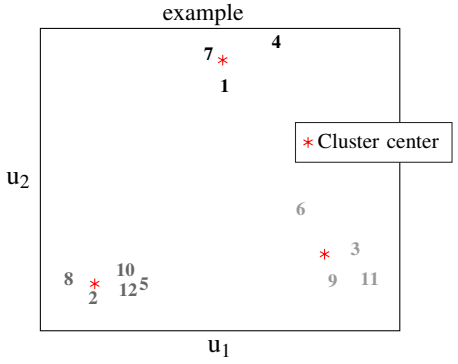
The sequence of data points to be measured is chosen through a random permutation of points from an arbitrary initial sequence. For all comparisons of the different strategies, ten of such random orders are generated and averaged.

E. Reasoning

The above proposed algorithms are based on the following considerations. Models relying only on small amounts of data are limited in their possible model accuracy. Through a clever placement of measuring points, the possible model accuracy can be influenced. The aim of all order determination strategies is to find subsets of points from the overall experimental design, that lead to the possibly best model qualities. Therefore a wide and nearly uniform coverage of the whole input space with small subsets is the goal. BGS tries to tackle the described considerations straight forward, filling the biggest gap of the input space with each additional point. One possible



(a) The intelligent k-means determines three clusters for the demonstration example



(b) The numbers represent the sequence determined by the IKMS procedure

Fig. 2: Explanation of the IKMS with the help of a demonstration example

weakness of BGS might be the exclusive concentration on points near the boundary at the beginning of the algorithm, especially in high-dimensional input spaces. This potential issue leads to the MDS algorithm, that tries to avoid only boundary-near points at the beginning. Adding the next point corresponding to the median distance instead of the maximum distance should lead to a better balance between points at the boundary and inside the input space for very small subsets. The IKMS algorithm exploits in a first step possibly existing structures in the experimental design by the intelligent k-means algorithm. All resulting clusters represent a specific area in the input space. Starting with all points next to these cluster centers should obtain information from all input space areas and in addition avoids only points at the boundary for very small subsets. It is somehow a mechanism to balance the exploration and exploitation of the whole experimental design. At the beginning all areas of the input space covered by the experimental design are explored and then successively exploited in more detail.

III. Comparison on Synthetic Functions

This section describes the experimental setup, that was chosen to compare the different strategies for the order determination based on synthetic test functions. A function generator is used to create random test functions for each input dimension p ranging from $p = 2$ to $p = 8$. Three different input distributions are used to generate training data sets for each random function and input dimension. Additionally, one huge test data set is generated for each random function and input dimension in order to assess the model quality. For each training data set the order of experimentation is determined according to the strategies proposed in Sec. II. Then, for training data increments of 10 % (in the determined order) a model is trained and its performance is assessed using the test data. Strategies for the order determination are compared based on how quick the model performance increases. Therefore results are averaged over all random functions created with the function generator. Here a big advantage of a function generator is exploited, i.e. designing synthetic functions with desired properties such as the input dimensionality, the amount of data and the data distribution, to name only a few.

A. Function Generator

The function generator described below is proposed in [9]. Basis for the used function generator are p-dimensional polynomials with M terms, that build a function $g(\cdot)$:

$$g(u_1, u_2, \ldots, u_p) = \cdots$$
$$\sum_{i=1}^{M} c_i \cdot (u_1 - s_{i1})^{q_{i1}} \cdot (u_2 - s_{i2})^{q_{i2}} \cdot \cdots \cdot (u_p - s_{ip})^{q_{ip}} \ . \quad (2)$$

All inputs u_i are assumed to be normalized, such that they vary in the unit hypercube $[0,1]^p$. The coefficients c_i are drawn from a normal distribution $N(0,1)$ and the shifts s_{ij} from a uniform distribution $U[0,1]$. Together with the normalized inputs, the latter ensures all the bases $u_j - s_{ij}$ to be in the range $[-1,1]$. Thus this range is kept after raising to the powers independent of the magnitude of the exponents q_{ij}. The powers q_{ij} are non-negative integer values. This is ensured by taking the floor of values drawn from an exponential distribution with expected value μ:

$$f(x|\mu) = \frac{1}{\mu} e^{-x/\mu} \ . \quad (3)$$

The expected value μ serves as a design parameter to control the probability of high exponents and the probability of high-order interactions, since exponents equal to zero make an input irrelevant for a certain term.

Since the usage of polynomial based random functions would favor model architectures that are at least partly based on polynomial approaches and the missing ability to mimic saturation characteristics, a transformation is proposed in [9]. The resulting polynomial $g(\cdot)$ is sent through a sigmoid function $h(\cdot)$:

$$h(u_1, u_2, \ldots, u_p) = \cdots$$
$$\frac{1}{1 + \exp\left(-\alpha \cdot g(u_1, u_2, \ldots, u_p)\right)} \ , \quad (4)$$

with tuning parameter α, controlling the amount of saturation.

B. Training and Test Data

In order to conduct the comparison, basically three different input distributions are used. Maximin Latin Hypercube (LH) designs, optimized according to the algorithm proposed in [10], data samples drawn from a uniform distribution and data samples drawn from two normal distributions with different mean values and equal standard deviations are employed. Figure 3 shows the different data distributions exemplarily in a two dimensional input space. It can be recognized, that the input coverage from the maximin LH design over data drawn from a uniform distribution down to the drawing from two normal distributions decreases. Additionally the chance to inhabit the same input twice or more often grows from Fig. 3a to 3c. These different properties are desired and should reveal distinct strengths and weaknesses from the order determination strategies. The number of training samples for all data distributions is kept constant at $N = 300$, while the number of inputs varies from $p = 2, \ldots, 8$. Due to the constant training samples and the increasing input dimension, different data densities arise.

In case of the uniform distribution and the two normal distributions 20 random test functions are used and 20 different instances (per input dimensionality) are generated for the comparisons. Because there is only one LH design (per input dimensionality), the number of test functions is increased to 100. For each input dimension the number of test samples is kept constant at $N_t = 1 \cdot 10^5$. The location of the test data samples is determined by a Sobol sequence [11] for each input dimension and is fixed for all synthetic functions.

C. Training Algorithm

The training algorithm used here is called HILOMOT (HIerachical LOcal MOdel Tree) and leads to the model class of local model networks (LMNs). The construction algorithm corresponds to the one presented in [12] and is based on the ideas of *hinging hyperplane trees*, which are described in [13], [14] and [15].

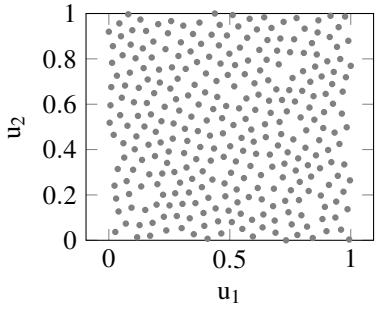
LMNs can be expressed in a basis function context, which means the model output $\hat{y}$ is calculated as sum of L so called local models $\hat{y}_i$ weighted with their validity functions Φ_i:

$$\hat{y} = \sum_{i=1}^{L} \hat{y}_i \Phi_i \ . \tag{5}$$

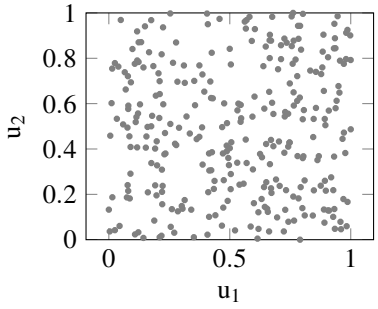
The validity functions describe the regions where the local models are valid; they represent the contribution of each local model to the output. For a reasonable interpretation of LMNs it is mandatory that the validity functions form a partition of unity:

$$\sum_{i=1}^{L} \Phi_i = 1 \ . \tag{6}$$

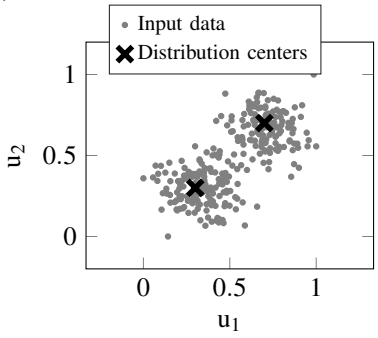
A training algorithm for LMNs has to determine the parameters of the validity functions - leading to the input space partitioning - and the parameters of the local models. HILOMOT typically uses local linear models. The validity regions are generated by sigmoid splitting functions, that are linked in a hierarchical, multiplicative way, see [12] for more details. The procedure of the HILOMOT algorithm can be explained with the help of Fig. 4. Starting with a global linear model, in each iteration an additional local linear model is generated. The local model with the worst local error measure is split into two submodels, such that the spatial resolution is adjusted in an adaptive way. The linear parameters of the new submodels are estimated locally by a weighted least squares method, nested in a nonlinear optimization for the determination of the splitting function parameters. In that way the separability of the nonlinear least squares problem is exploited as recommended by [16], leading to a computationally very efficient solution. A major advantage of HILOMOT is the possibility to perform axes-oblique splits, which makes this training algorithm very suitable for high-dimensional input spaces. This is achieved



(a) Maximin optimized LHS



(b) Data drawn from uniform distribution



(c) Data drawn from 2 normal distributions

Fig. 3: Data used for the comparison of different order determination strategies

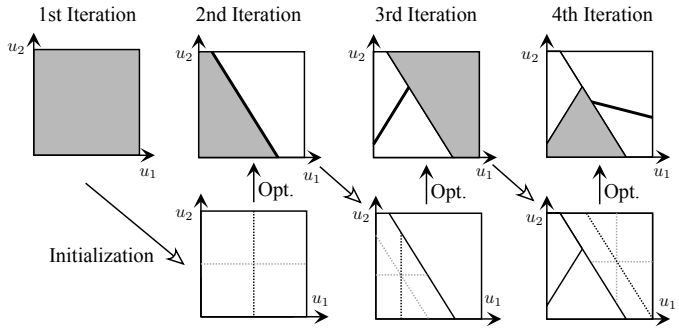


Fig. 4: Procedure of the HILOMOT algorithm for the first 4 iterations partitioning a 2-dimensional input space

by optimizing the current split direction and position in each iteration. Only the new split is optimized, all already existing splits are kept unchanged. The initial split direction for the optimization is either one of the orthogonal splits or the direction of the parent split.

D. Synthetic Function Comparison Results

The achieved test errors for an increasing amount of training data are presented in Fig. 5 for all three data distributions and input dimensions $p = 2$, $p = 5$ and $p = 8$. Each column corresponds to one input dimensionality, each row to a data distribution. Most diverse results are yielded in case of Fig. 5g, where the experimental design is drawn from two normal distributions and the input dimensionality is $p = 2$. In this case the IKMS turns out to be the best method, since the resulting test error is for almost all amounts of training data the lowest. The BGS method lays in between random sequences and the MDS procedure, which is the worst performing method. With an increasing input dimensionality, the benefit of IKMS vanishes and the test errors of all methods get closer to each other. For input distributions yielding a better coverage of the input space, all procedures perform equally well except for MDS, which at least for two dimensional input spaces turns out to be the worst method. In summary, IKMS seems to be the most promising approach for the order determination. Significant advantages can be observed for input distributions that incorporate a specific structure, while no drawbacks for uniformly covered input spaces are visible.

IV. Comparison on a CFD Metamodel

This section compares the effect of different order determination strategies on the quality of CFD metamodels. CFD (Computational Fluid Dynamics) is a numerical method to solve the Navier-Stokes equations which describe the state of a flow field. The training and test data used in this section is identical to the CFD dataset used in reference [17] which discusses metamodels for the prediction of the specific pressure rise (ψ) and the efficiency (η) of centrifugal fans. ψ and η represent the metamodel outputs. They depend on nine geometrical input parameters and one operational input parameter (the specific flow rate Φ). An optimized LH design according to [10] is used to generate sets of geometrical inputs. For each geometry variation, numerous specific flow rates are simulated by CFD and the corresponding outputs are stored in the dataset. Altogether the dataset contains the results of 5528 CFD simulations consisting of 928 geometry variations and an averaged number of approximately 6 flow rate variations for each fan geometry. All details regarding the gathering of the data (e.g. a full description of the CFD model) can be found in [17].

A. Comparison Procedure

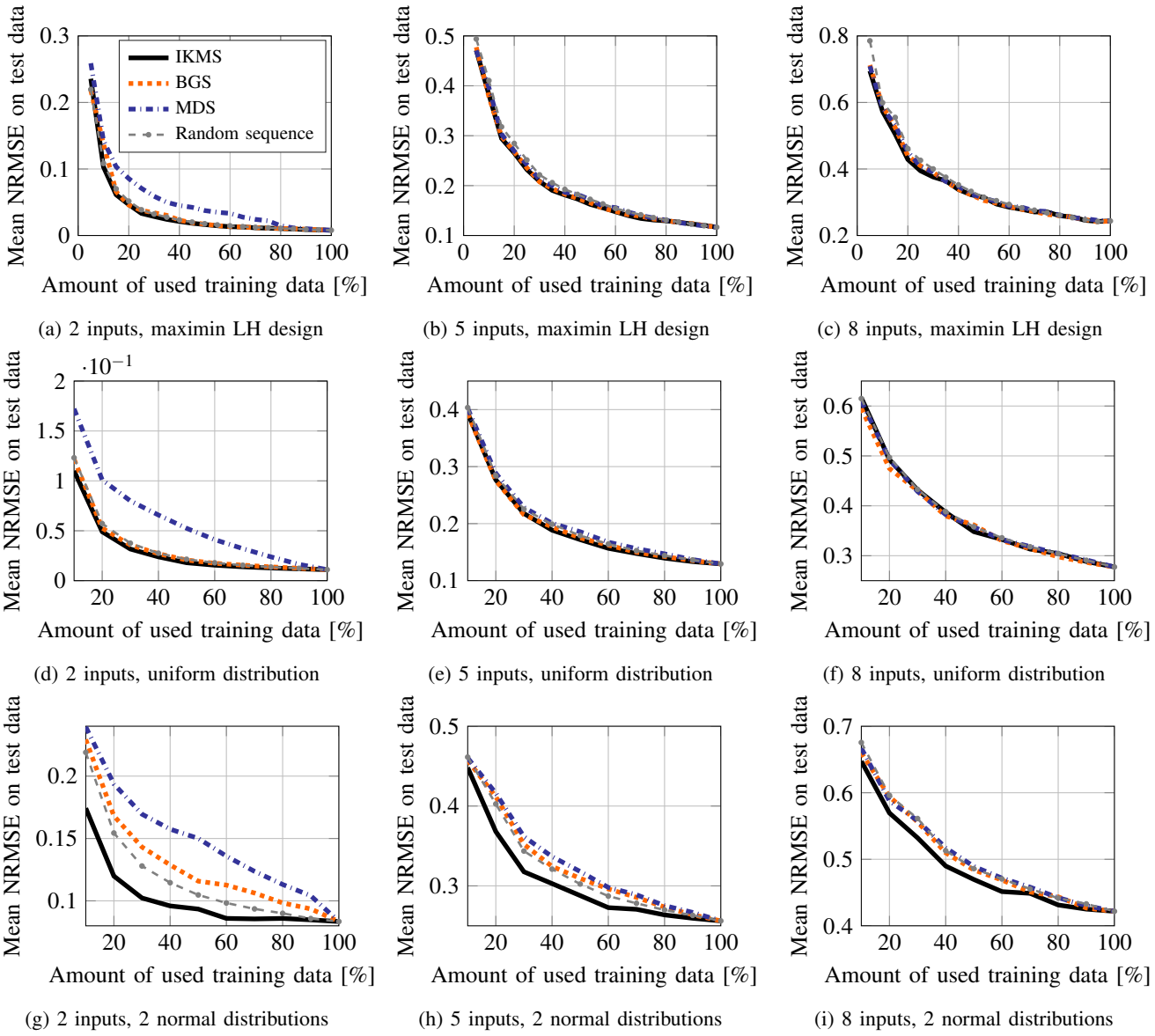
In order to compare all order determination strategies on a fixed data set, where it is not possible to generate an arbitrary amount of test data, the determination of the test errors differs from the method described in Sec. III. All available data is randomly split into 85 % for training and 15 % for testing purposes. This is done 50 times, such that there are 50 different training and test data sets. Measurement sequences are only determined for the training data sets. Again, in steps of 10 % of the training data, models are generated with HILOMOT (see Subsec. III-C) and the error on the corresponding test data is evaluated. As a result, 50 different test errors are obtained for each amount of training data. The mean value of all 50 normalized root mean squared errors (NRMSE) on test data serves as comparison criterion.

B. CFD Metamodel Comparison Results

Figure 6a and 6b show the mean NRMSE values on test data versus the percentage of training samples for the prediction of ψ and η respectively. Comparing all algorithms, IKMS and BGS perform equally well whereas MDS has the slowest increase in terms of model quality. Now, only the worst and best case of the randomly chosen orders of experimentation are shown. For each of the 50 data splitting realizations, ten random sequences are generated, such that there are 500 different random sequences in total. Showing the best and worst case should provide a feeling how good or bad the strategies perform on an absolute scale. Note that the NRMSE values of the worst and best random order do not coincide with the NRMSE values of the other determination strategies at 100 % of the used training data. This is due to the fact, that each of the two shown random sequence curves is generated with only one of the 50 resulting training data sets. The IKMS and BGS curves are very close to the best case of all random sequences for all amounts of used training data. Therefore, these strategies are not only performing well compared to the other strategies, but also on an absolute scale.

V. Conclusions

Models can already be used, while the measurement process is still in progress. The accuracy of models in early stages of the measurement process highly depends on the order, in which the measurements are carried out. This paper presents several strategies for the order of experimentation determination given an already existing experimental design for regression problems in metamodeling tasks. With the help of a function generator, the data distribution and the data density

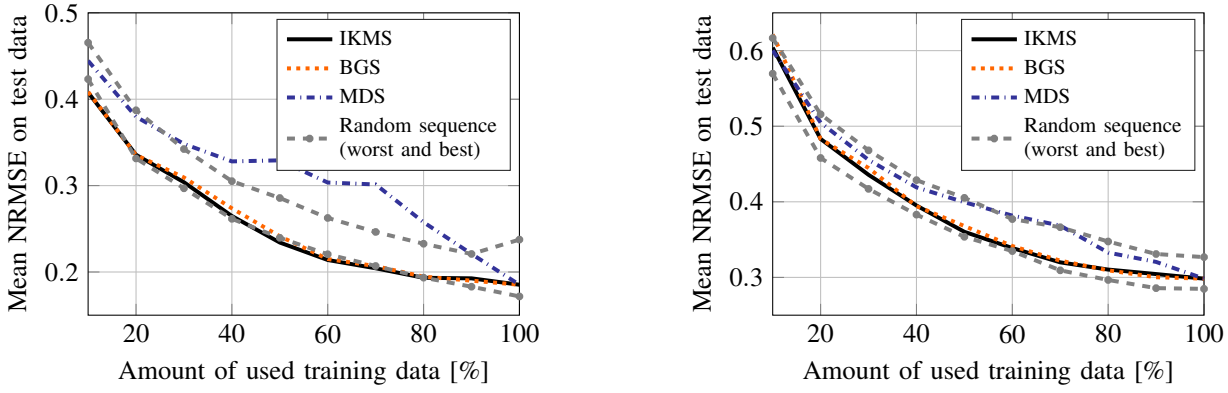


(a) 2 inputs, maximin LH design (b) 5 inputs, maximin LH design (c) 8 inputs, maximin LH design

(d) 2 inputs, uniform distribution (e) 5 inputs, uniform distribution (f) 8 inputs, uniform distribution

(g) 2 inputs, 2 normal distributions (h) 5 inputs, 2 normal distributions (i) 8 inputs, 2 normal distributions

Fig. 5: Mean NRMSE on test data versus the amount of training data for different input dimensions and distributions



(a) ψ prediction (b) η prediction

Fig. 6: Comparison of different order determination strategies for the prediction of ψ and η

is varied for several randomly generated synthetic functions. It is shown, that especially dense and structured input data benefits from well ordered measurements. For sparsely and uniformly covered input spaces almost all presented ordering strategies perform equally well. In sum, the proposed IKMS strategy yields the best results on synthetic functions.

The applicability to real metamodeling tasks is proven with the help of an existing data set for a CFD metamodel. The BGS and the IKMS strategy perform equally well and their average performances are near the best out of 500 randomly generated orders. Considering the results from the synthetic functions together with the ones from the CFD metamodel, the IKMS strategy is recommended to be used.

With the help of the proposed order determination strategies it is possible to enhance the accuracy of models in early stages of the measurement process. Especially in combination with active learning strategies, where additional measurement queries are determined based on models trained with the currently available data, improvements can be expected.

For the application on real-world test benches additional considerations are necessary. The proposed strategies are all based on distance measures. If there are hard-to-vary factors on the test bench, a specific input weighting can be used to influence the algorithms. Through the input weighting, points appear closer or farther away, such that sequences can be generated, where hard-to-vary factors are only changed slightly going from point to point of the ordered list. In a similar way the concept of blocking can be realized. The concept of randomization is somehow already included in the favored methods, i.e. IKMS and BGS, since biggest gaps are filled. As a result, points that are close to each other in the input space will be far away in the ordered lists generated by IKMS and BGS. Thus, the time between the measurements of two similar measuring points is increased and effects resulting from factors, that can not be controlled, can average out.

VI. Acknowledgments

This work was funded by the German Ministry for Economic Affairs and Energy (BMWi), the German Federation of Industrial Research Associations (AiF) and the Research Association for Air and Drying Technology (FLT).

References

[1] A. A. Correa, P. Grima, and X. Tort-Martorell, "Experimentation order in factorial designs: new findings," *Journal of Applied Statistics*, vol. 39, no. 7, pp. 1577–1591, 2012.

[2] H. Hilow, "Comparison among run order algorithms for sequential factorial experiments," *Computational Statistics & Data Analysis*, vol. 58, pp. 397–406, 2013.

[3] T. Tambouratzis, "Counter-clustering for training pattern selection," *The Computer Journal*, vol. 43, no. 3, pp. 177–190, 2000.

[4] B. Hartmann, T. Ebert, and O. Nelles, "Model-based design of experiments based on local model networks for nonlinear processes with low noise levels," in *American Control Conference (ACC), 2011*. IEEE, 2011, pp. 5306–5311.

[5] S. Shan and G. G. Wang, "Metamodeling for high dimensional simulation-based design problems," *Journal of Mechanical Design*, vol. 132, p. 051009, 2010.

[6] M. Meckesheimer, A. J. Booker, R. Barton, and T. Simpson, "Computationally inexpensive metamodel assessment strategies," *AIAA journal*, vol. 40, no. 10, pp. 2053–2060, 2002.

[7] T. J. Santner, B. J. Williams, and W. Notz, *The design and analysis of computer experiments*. Springer, 2003.

[8] B. Mirkin, *Clustering for data mining: A Data Recovery Approach*. Chapman & Hall/CRC, London, 2005.

[9] J. Belz and O. Nelles, "Proposal for a function generator and extrapolation analysis," in *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on*. Madrid, Spain: IEEE, Sep. 2015, pp. 282–287.

[10] T. Ebert, T. Fischer, J. Belz, T. Heinz, G. Kampmann, and O. Nelles, "Extended deterministic local search algorithm for maximin latin hypercube designs," in *IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, Cape Town, South Africa, Dec. 2015.

[11] H. Niederreiter, "Low-discrepancy and low-dispersion sequences," *Journal of number theory*, vol. 30, no. 1, pp. 51–70, 1988.

[12] O. Nelles, "Axes-oblique partitioning strategies for local model networks," in *IEEE International Symposium on Intelligent Control*, 2006, pp. 2378–2383.

[13] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," *Information Theory, IEEE Transactions on*, vol. 39, no. 3, pp. 999–1013, 1993.

[14] S. Ernst, "Hinging hyperplane trees for approximation and identification," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 2. IEEE, 1998, pp. 1266–1271.

[15] S. Töpfer, "Approximation nichtlinearer Prozesse mit Hinging Hyperplane Baummodellen (Approximation of nonlinear processes with hinging hyperplane trees)," *at-Automatisierungstechnik*, vol. 50, no. 4/2002, p. 147, 2002.

[16] H. B. Nielsen, "Separable nonlinear least squares," Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Tech. Rep., 2000.

[17] K. Bamberger, J. Belz, T. Carolus, and O. Nelles, "Aerodynamic Optimization of Centrifugal Fans Using CFD-Trained Meta-Models," in *16th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC)*, April 2016.