

Tool-assisted Surrogate Selection for Simulation Models in Energy Systems

Stephan Balduin*, Frauke Oest*, Marita Blank-Babazadeh*, Astrid Nieße[‡] and Sebastian Lehnhoff*

* OFFIS – Institute for Information Technology, Oldenburg, Germany

[‡] Leibniz University Hannover, Germany

Email: {frauke.oest, stephan.balduin, sebastian.lehnhoff, marita.blank-babazadeh}@offis.de
niesse@ei.uni-hannover.de

Abstract—Surrogate models have proved to be a suitable replacement for complex simulation models in various applications. Runtime considerations, complexity reduction, and privacy concerns play a role in the decision to use a surrogate model. The choice of an appropriate surrogate model though is often tedious and largely dependent on the individual model properties. A tool can help to facilitate this process. To this end, we present a surrogate modeling process supporting tool that simplifies the process of generation and application of surrogate models in a co-simulation framework. We evaluate the tool in our application context, energy system co-simulation, and apply it to different simulation models from that domain with a focus on decentralized energy units.

I. INTRODUCTION

THE simulation of smart grids is a key step in the deployment process of new technologies and methodologies in the present power system for safety and costs reasons. Co-simulation frameworks like mosaik¹ assist the simulation process in the energy domain by providing programmable interfaces for different simulation models and realizing data flow dependencies including synchronization issues. These simulation models can become quite complex and can be provided in different programming languages. This can lead to a slowed down performance of a smart grid simulation. Performance plays a role especially in large-scale setups, such as in the research projects Smart Nord [1] or D-Flex [2], which are required for the evaluation of new Smart Grid algorithms or sustainability assessments. Furthermore, simulation models might be supplied by industrial stakeholders and thus must be considered as intellectual property that should not be disclosed to partners.

A solution concerning these issues might be the use of a data-driven abstraction of simulation models, so called surrogate models. A surrogate model is a function that maps input values to one or more output values. For this purpose, machine learning algorithms can be used to determine the relation between input and output by training with sample data generated by the original simulation model [3]. The creation of those surrogate models underlies several degrees of freedom like the choice of a sampling strategy for the input data and the choice of the surrogate algorithm. The performance of a surrogate for a particular simulation model

depends not only on the specific type of that model but also can be measured differently depending on the evaluation function. An evaluation function measures the similarity of outputs between surrogate and simulation model for a set of input combinations. Based on the number of existing surrogate modeling algorithms, the identification of an appropriate surrogate can be computationally quite intensive and should be (semi-)automated to ensure replicability and comparability of the results.

For these reasons we propose to use a tool to support the selection of appropriate surrogate models. With the help of various evaluation functions, we can then evaluate the performance of different settings from specific sampling strategies and surrogate models. Furthermore, we present the Python-based open source tool MeMoBuilder² to support this process. MeMoBuilder provides a semi-automated surrogate modeling process including comparison with the original simulation model in a time series evaluation.

The rest of this paper is structured as follows. First, in Chapter II we present the surrogate modeling process and highlight the challenges that emerge from this. In Chapter III we look at existing tools and other work in the field of surrogate modeling. Chapter IV focusses on the tool MeMoBuilder which is our approach for reducing the complexity of the surrogate modeling process. In Chapter V we will present a case study to evaluate the tool with reflection to the defined challenges. This paper ends with conclusion and outlook on future work in Chapter VI.

II. CHALLENGES IN SURROGATE MODELING

The process of creating a surrogate model, which is also called meta model or response surface, is well documented in the literature, e. g. in Myers et al. [4]. In the following, we briefly recap the surrogate modeling process as described by Forrester et al. [5] to point out the challenges in this process. We then derive the requirements that are important for a surrogate modeling tool.

A. Choosing the sampling strategy

The first step of this process starts with the generation of so-called samples through sampling strategies. We use

¹<http://mosaik.offis.de/>

²<https://github.com/stbalduin/memobuilder>

the term sampling strategy for the application of a sampling design, i.e. the theoretical construction to generate samples. These consist of a set of possible input combinations $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$ for the original simulation model $f(\mathbf{x})$ and their corresponding outputs \mathbf{y} calculated by the simulation model. A sampling strategy defines which input combinations will be chosen to generate samples and it is important to pick the most relevant data to generate a good model. Finding an appropriate sampling strategy can depend on certain problem-specific and often contrary requirements, e.g. being deterministic, well-balanced, guaranteeing to cover the whole sampling space, or to work well with relatively few samples [6]. The goal of these strategies is to cover the relevant sample space, i.e. non-trivial (e.g. non-linear) behavior of the simulation model is included as accurate as possible. A well-balanced sampling design can be generated deterministically, but may require a large number of samples. On the other hand, a non-deterministic sampling design may work well with fewer samples, but there is no guarantee that it will cover the whole sample area. In both cases the orthogonality, i.e. the correlation of inputs, has to be considered. Simpson et al. [6] point out that the information gain of a design is balanced against the cost of experimentation, i.e. the number of samples, and lists several measures of merit which are useful to compare designs.

B. Choosing the surrogate algorithm

In the following step, the surrogate model $\hat{f}(\mathbf{x})$ is created by applying so called surrogate algorithms on the previously generated data. We use the term surrogate algorithm for any supervised machine learning algorithms that is capable of creating a surrogate model. A set of samples is used as training data (\mathbf{X}, \mathbf{y}) for the surrogate algorithm to adjust its parameters in order to make the resulting surrogate model as similar as possible to the original simulation model. The surrogate algorithm can be picked from a large variety of machine learning algorithms with trade-offs in their characteristics, e.g. suitability for non-linear problems, suitability for high-dimensional data, complexity in the application, or in the learning phase. The latter is often partially depending on the search for optimal hyperparameters. These kind of parameters have to be set a priori to the learning process. To find the most appropriate parameter values, an exhaustive searching process with cross-validation has to be applied which means that different splits of the samples into training set and test set are evaluated. Furthermore, some surrogate algorithms use kernel functions to build the surrogate model. The choice of kernel functions and the hyperparameter tuning of these functions also have to be optimized to the given problem.

C. Choosing the evaluation function

To evaluate the quality of the surrogate model an evaluation function is used. In general, the error ϵ is used to describe the deviation between original simulation model and surrogate model: $f(\mathbf{x}) = \hat{f}(\mathbf{x}) + \epsilon$, but there are also evaluation functions that represent the quality of the model in a different way,

e.g. correlation functions. The quality of the surrogate model approximation can be evaluated by using samples as test data (which should be distinct from the training data) on the surrogate model and as well on the original simulation model. The error ϵ can be determined by applying an evaluation function on both resulting outputs. The choice of the evaluation function has a strong influence on the ranking of surrogate models. They can be categorized into optimistic and pessimistic functions [7]. An optimistic evaluation function weights small errors more than large ones, hence might be beneficial if the error fluctuates greatly. Pessimistic evaluation function behaves the other way around, therefore they might be useful if large errors are undesirable. But there are also other characteristics, e.g. interpretability and independence of (physical) units, which should be taken into account when selecting an evaluation function for the model. It is important to know which requirements the model has to fulfill, like how critical small errors are or who will use the model afterwards, to decide which criteria should be prioritized.

D. Requirements for tool support

Several degrees of freedom can be found in the defined process steps of surrogate model creation and evaluation, namely the choice of the sampling strategy, the choice of the surrogate algorithm, and the choice of an evaluation function used to evaluate this model. Each choice has its benefits and drawbacks. To find the most suitable combination is always depending on the given problem and cannot be generalized [5, p. 18]. For this reason, several iterations of sample generation, model creation, and model evaluation need to be performed during the surrogate modeling process until the results meet the requirements of the application context. The multi-dimensionality of the surrogate modeling process is summarized by Figure 1. For multiple but similar structured

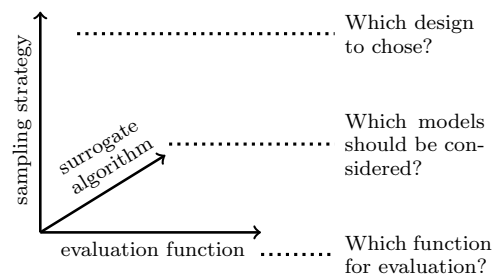


Fig. 1. Dimensions of the surrogate modeling process. Each dimension itself can be optimized quite easily, but it becomes a trade-off when all of these aspects shall be taken into account for optimization.

simulation models this is a quite repetitive process when modeling surrogates, so a tool for assistance is strongly recommended. Such a tool should fulfill the following requirements:

- R1 Support the surrogate modeling process: The tool should allow the surrogate modeler to address all degrees of freedom in experimental design, choice of surrogate algorithm, and evaluation function and thus allow for an

application specific instantiation of the surrogate modeling process.

- R2 Facilitate model exchange: The surrogate modeler is not necessarily the person who will use the model afterwards. Therefore, the tool should allow to create surrogate models which can be easily integrated into an existing environment (e.g. a co-simulation framework) and replace the original simulation models in order to easily integrate these models into smart grid simulation scenarios.
- R3 Allow modularity: In some setups, sampling data may be retrieved from other sources than available simulation models, e.g. in industry driven studies. To allow the surrogate modeler to perform only parts of the process, the tool should support a separation of concerns so that the integration of simulation models for sampling and the construction of surrogate models are independent of each other.

Although not a specific requirement in the choice or development of an appropriate surrogate modeling tool, the long-term perspective of using such a tool should be the (semi-)automatic generation of surrogate models.

III. RELATED WORK

The whole surrogate modeling process is targeted by the Matlab Surrogate Modeling (SUMO)-Toolbox³ which automates the single steps of this process. The SUMO-Toolbox builds a surrogate model of a given data source and needs only a few configurations by the user like accuracy and time constraints. However, the resulting surrogate model is bound to the Matlab environment. To deploy the model in a different setup, adaptations may be required and therefore requirement R2 is not fulfilled.

A tool aided surrogate selection is described by Mehmani et al. [8]. The authors developed the Concurrent Surrogate Model Selection (COSMOS) Framework which can be used to select an appropriate surrogate model. This tool focuses on the surrogate selection itself which comprises the optimal model type, the optimal kernel function (if needed), and optimal values of hyperparameter (if present). Despite being an important contribution in the domain of surrogate modeling, some shortcomings arise with respect to requirement R1: An easy comparison of different experimental designs is not possible.

Although not applicable to the problem of generating a surrogate for a given simulation model, the Waikato Environment for Knowledge Analysis⁴ (Weka) proposed by Hall et al. [9] is an important open source collection of machine learning algorithms which aims to make these algorithms generally available to solve practical problems. Weka provides both a programmable and a graphical interface where no programming skills are needed when a learnable dataset is given. The focus of this tool is on data mining. Therefore, it does not contain an interface to integrate a simulation model

and automatically generate learnable data. Nonetheless, it does not support to address all degrees of freedom of the surrogate modeling process (requirement R1).

Various applications of surrogate models can be found in Koziel et al. [10], though there are no applications in the energy domain. Other works deal with the construction of a surrogate model for specific (simulation) models within concrete use cases. Pinto et al. [11] constructed a surrogate model for multi-period flexibility provided by a home energy management system. They modeled local microgeneration units, like photovoltaics, combined with flexible storage equipment which can be a battery. In their study the authors proposed an algorithm based on evolutionary particle swarm optimization to generate feasible flexibility trajectories. These trajectories were successfully used as training data for a support vector data description (SVDD) machine learning algorithm.

	SuMo Toolbox	COSMOS	Weka
R1 (Modeling process)	✓	✗	✗
R2 (Model exchange)	✗	✗	✗
R3 (Modularity)	✓	✗	(✓)

TABLE I
SUMMARY OF THE PRESENTED TOOLS COMPARED TO OUR REQUIREMENTS.

In our research (see Table I) we did not find a tool that fulfills all requirements as defined in chapter II.

IV. MEMOBUILDER

In this chapter we describe the architecture of the proposed Meta Model (MeMo) Builder. Our goal was to integrate surrogate algorithms, sampling strategies, and evaluation functions into one tool (requirement R1). This tool selects the optimal from each of those and generates a surrogate model that can be used within a co-simulation framework (requirement R2) as a replacement for the original simulation model. The model should also be compared with the original simulation model and behave similarly to it. For co-simulation we choose mosaik since it is a flexible tool which provides interfaces to models written in different programming languages. Thus, the surrogate modeling process is applicable regardless of the model at hand and the modeler can concentrate on the modeling process itself rather than integrating the model. The surrogate model itself can also be easily integrated in mosaik.

Mosaik facilitates a time discrete simulation, i.e. each simulation step has the same fixed length and each simulator can decide when it will be activated. Simulation models used in such a framework need to perform their simulation step for a given time interval, and a defined set of inputs and parameters as shown in Figure 2. The same applies to the surrogate models we want to build. We developed the MeMoBuilder as a prototype to identify challenges and benefits of the surrogate modeling process for simulation models in energy system, and adapt the tool to the needs identified in following this process. Further, we wanted to explore the possibilities of (semi-)automatic surrogate model generation of mosaik

³<http://sumo.intec.ugent.be/>

⁴<http://www.cs.waikato.ac.nz/ml/weka/>

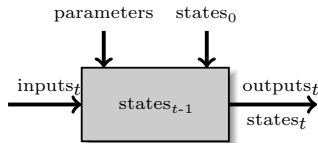


Fig. 2. The simulation model is initialized with parameters and initial states₀. When simulation starts the model gets inputs_t for step *t*. The results of each step are the states_t which will be saved internally and then used in step *t* + 1, and outputs_t.

component models for power and smart grid simulation scenarios. MeMoBuilder provides a set of sampling strategies, surrogate algorithms, and evaluation functions which can be chosen to generate a surrogate model. In Figure 3 the modular

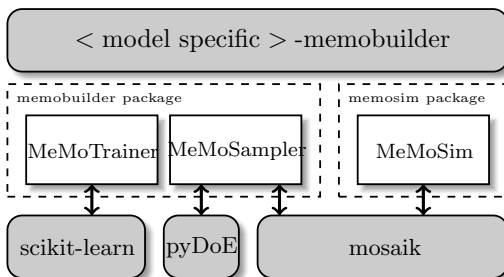


Fig. 3. MeMoBuilder has a modular structure where single components can be left out or be replaced. The core packages are the MeMoSampler and the MeMoTrainer which only need to be adapted if different frameworks for machine learning or design of experiments are used. The MeMoSim package depends on the chosen co-simulation framework which is in this case mosaik.

architecture (requirement R3) and main components of the MeMoBuilder are shown. For each simulation model, a model-specific MeMoBuilder is configured in a YAML⁵ (YAML Ain't Markup Language) configuration file. This configuration file is also used for other degrees of freedom like how and which of the other MeMoBuilder components will be used in the surrogate generation process.

In the first step of this process the MeMoSampler uses a framework like pyDoE⁶ to generate the sampling designs configured in the YAML file. According to these designs, the simulation model is sampled within a mosaik scenario to create one or more training sets. The use of mosaik at this point ensures that the MeMoSampler can be applied to every model with an existing mosaik adapter regardless for which simulation environment it was built originally.

Next, the training sets are used to create surrogate models. Thereby, it depends on the configuration how many surrogates will be generated. Each training set is used by the MeMoTrainer for all surrogate algorithms that are configured in the model-specific MeMoBuilder. The MeMoTrainer itself uses the scikit-learn library⁷ [12] for model fitting and cross validation of optimal hyperparameters, but other frameworks could be integrated as well. It is also possible to use multiple

evaluation functions and in this case MeMoTrainer generates a surrogate for each function.

Ultimately, there can be a whole set of surrogates and each of these will be rated using different evaluation functions. When the surrogate model generation is finished, MeMoBuilder compares the simulation model with the surrogate model in a simple simulation scenario within the chosen co-simulation framework mosaik. The results of each simulation step are stored in a database. Additionally, a visualization of these results is generated and stored. Once the simulation is finished, it is possible to see differences in the output behavior between simulation model and surrogate model.

V. CASE STUDY

To test the functionality of the MeMoBuilder in a practical environment and check the suitability of this tool against the requirements as defined in Section II, a selection of surrogate algorithms, sampling strategies, and evaluation functions was identified and integrated in the MeMoBuilder environment.

A. Chosen sampling strategies

We integrated four sampling strategies of both random and deterministic type.

a) *Random based sampling strategies:* Latin Hypercube Sampling (LHS) is probably the most common strategy and has some advantages which possibly lead to the wide acceptance of this method: It is well balanced while only a small number of samples is needed [3, p. 198ff]. LHS is nearly as easy to apply as the other random-based strategy we use, the Monte Carlo Sampling (MCS) which is pure random selection of sample points. Both give the chance, but not the guarantee that the whole sample space will be covered. Furthermore, in some setups the orthogonality of these designs is not given [13, p. 42].

b) *Deterministic sampling strategies:* Besides the random based strategies, we used two deterministic strategies. Both, the sequence of Halton (HSEQ) and the sequence of Sobol (SSEQ) use prime numbers to generate a sequence of numbers, e.g. the prime number 2 generates the sequence $\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \dots$. While HSEQ varies the prime numbers to generate a new sequence, SSEQ permutes this sequence using primitive polynomials. For a more detailed explanation we refer to Lemieux [14, 157ff]. The sample space generated by deterministic designs is typically well-balanced and has only occasionally issues with missing orthogonality [13].

B. Chosen surrogate algorithms

As surrogate algorithms we selected five heterogeneous algorithms from the field of interpolation, neural networks, and other regression methods. One of the simplest surrogate algorithm is linear regression such as LASSO which constitutes a fast polynomial approximation According to Hastie et al. [15, p. 43] this regression function can outperform more complex methods if the data is structured linearly, a small set of training data, or sparse data is used.

Another approach is to use lazy learners like the k-nearest neighbors (k-NN) algorithm. As described in Yang et al. [16]

⁵<http://yaml.org/>

⁶<https://pythonhosted.org/pyDOE/>

⁷<http://scikit-learn.org/>

the k nearest neighbors of the learned samples are directly used to estimate missing outputs. According to Ertel [17, p. 199], apart from finding the correct hyperparameters, k -NN has no actual learning phase therefore it belongs to the lazy learners. For each estimated output, k -NN calculates the distance of each sample to find the k nearest samples. Ertel also points out finding the next nearest neighbors according to the given input can be computational intensive if many training samples are used. According to Samaniego and Schulz [18] its strength lies in the flexibility which makes k -NN an appropriate choice for non-linear data structures [19].

According to Cui et al. [20] Kriging is often used to interpolate data between known data points which is done by a combination of a polynomial model and a realization of a normally distributed Gaussian random process. Simpson et al. [6] state that the strength of Kriging lies in the variety of correlation functions that can be used to shape the Gaussian random process.

Support Vector Regression (SVR) is a type of support vector machines with similarities to Kriging, since the heart of both is a kernel function [21]. This algorithm uses the kernel function in order to transform non-linear regression problems into linear by mapping the original input space to a higher feature dimension space [22].

An ANN is made of several interconnected neurons which process data coming either from outside or from other neurons. The challenge in creating ANN focuses on architectural design and the number of neurons that should be used [6]. A well constructed ANN can be quite powerful in this sense that they can handle non-linear data structures [23] as well as they can handle high-dimensional data, although this can be computationally intensive. A mitigation of computing time could be achieved by parallel computing [6]. For this purpose, multi-layer perceptron regression will be used.

C. Chosen evaluation functions

To evaluate the generated surrogate models diverse evaluation functions were selected. As a pessimistic evaluation function we choose the mean squared error (MSE) where outliers are weighted quadratically in order to punish large errors more than small errors. The mean absolute error (MAE) is punishing outliers linearly, so it is less pessimistic than the MSE. It is easier to interpret than the MSE since units are not effected by this function.

We also choose the determination function R^2 which is related to the Pearson Correlation Coefficient [3, p. 113]. In contrast to error functions where the error should be minimized, in R^2 a value close to 1 means a high correlation of data and therefore the surrogate is similar to the original model. Hence, an R^2 close to 0 or even negative values can be interpreted as low correlation and thus the surrogate model is not well modeled. The R^2 is free of units which leads to intuitive interpretations of this function. The evaluation functions described above are taken from the scikit-learn library. More information about these functions can be found in their

documentation and user guides⁸. In addition to the scikit-learn evaluation functions, we choose the harmonic average error (HAE), which is shown in Equation 1.

$$\text{HAE}(y, \hat{y}) = \left(\frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{\sqrt{(y_i - \hat{y}_i)^2}} \right)^{-1} \quad (1)$$

Here, the n is the number of samples, y the result of the original model, and \hat{y} the result of the surrogate model as it is the approximation of the original model. This function allows to dominate small errors over large errors which means that this metric allows to have few large errors if there are small errors to compensate. Therefore, the HAE is considered to be optimistic.

D. Chosen simulation models

We conduct our case study using three simulation models representing different home energy system units that were already in use for different energy system simulation scenarios. These models will be briefly explained in the following without going to much into detail.

The first model is a battery which has an internal state of charge and takes the target electrical power as input. In each step, the electrical power output is calculated depending on the current state of charge. The output has a negative sign if the battery "consumes" energy, otherwise the output has positive sign.

The second model is that of a photovoltaic (PV) plant system. The model has the module temperature as internal state and uses several input variables like time stamp, solar radiation, and air temperature. Based on geo and other information stored in the model, the sun position is calculated depending on the current time stamp which is then used to compute the electrical power output depending on current radiation on the surface of the PV plant.

The last model is the fuel cell (FC) which produces power and heat at the same time. We consider electrically driven operation, i.e. the FC follows a certain power output rather than a thermal profile in thermally driven operation. This model has two inputs, three outputs, and the electrical power as internal state. The inputs are the temperature of the incoming heating water and the target electrical power for the next time interval. The outgoing temperature of the heating water, the thermal power, and the actual electrical power are regarded as outputs of the model. The actual electrical power is divided into discrete fixed electrical power stages whereas the thermal outputs can have continuous values. It should be denoted that the actual electrical power does not need to be the same as the targeted electrical power due to internal restrictions of the model.

All models were build at OFFIS and for each of them we used MeMoBuilder to generate a surrogate model for all combinations of the mentioned methods which results in $4*5*4 = 80$ surrogate models. We used a uniform sample size of 5,000.

⁸https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

MeMoBuilder implements methods to apply cross-validation and hyperparameter optimization on this sample size. Since it is easily interpretable, we picked the best surrogate according to the R^2 score [7]. This surrogate model will be compared with the original model in a simulation scenario.

E. Results for the battery model

The best five surrogate models for the battery are shown in Table II. Note that these are the best models according to the R^2 . Using a different score for sorting may result in a different order of the models. For our battery model, the best combination consists of a Latin Hypercube sampling strategy, an artificial neural network, and the mean squared error for evaluation. But the MeMoBuilder also provides the results for the other combinations. In case of the battery model we see that support vector regression would as well be an appropriate surrogate model. Next, a comparison in a simulation setup is

Sampling Strategy	Surrogate Algorithm	Train Func.	R^2 Score	HAE Score	MAE Score	MSE Score
LHS	ANN	MSE	0.998	$2.67 \cdot 10^{-3}$	0.024	0.005
HSEQ	SVR	MSE	0.992	$1.5 \cdot 10^{-10}$	6.535	873.0
MCS	SVR	MSE	0.992	$3.76 \cdot 10^{-3}$	5.872	802.0
MCS	SVR	HAE	0.992	$2.78 \cdot 10^{-3}$	5.909	750.5
LHS	SVR	MSE	0.991	$2.27 \cdot 10^{-2}$	6.766	863.2

TABLE II

BEST FIVE SURROGATE MODELS OF THE BATTERY ACCORDING TO THE R^2 SCORE. THE COMPARISON USING DIFFERENT SCORES REVEALS THAT SVR SCORES POORLY DESPITE A VERY GOOD R^2 SCORE WHEN THE MSE IS CONSIDERED.

done. Both models are supplied with a schedule of electrical power targets. The results are plotted in Figure 4. We see for the electrical output (P_{el}) the surrogate model is quite accurate most of the time. Only at the last part the value seems to oscillate. For the internal state of charge, the surrogate model results are accurate for the first 30 - 35 steps. At that point, the target power value is set to zero which is not correctly handled by the surrogate model. After that point the deviation of the prediction increases. At about step 150 the state of charge of the surrogate model reaches zero which may be the reason for the oscillating power output of the surrogate model.

The results show the difficulties of modeling internal states which converge to certain boundaries like minimal and maximal state of charge. Therefore, we could use the MeMoBuilder to investigate further combinations, e. g. other sampling strategies as the peripheral areas of the sample space seem to be not sufficiently covered by the Latin Hypercube strategy, but this is beyond the scope of this paper.

F. Results for the photovoltaic plant

In Table III the best five surrogate models for the PV plant are shown. The best combination in this case is a Latin Hypercube sampling, an ANN, and the R^2 score, but the same combination with a Monte Carlo sampling differs only very slightly after the decimal point. However, in this case the LHS

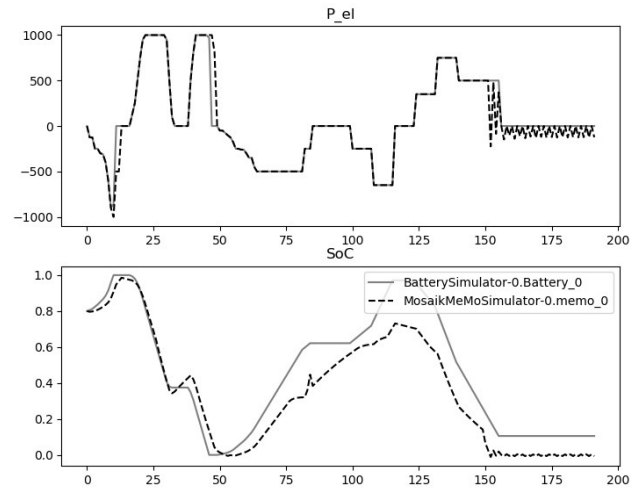


Fig. 4. Co-simulation of surrogate and original battery model comparing their electric power output and the state of charge. The grey line is the original model and the black dashed line is the surrogate model.

model is better not only at R^2 , but also according to MAE and MSE scores. In the simulation, both models are provided

Sampling Strategy	Surrogate Algorithm	Train Func.	R^2 Score	HAE Score	MAE Score	MSE Score
LHS	ANN	R^2	1.0	$2.21 \cdot 10^{-2}$	0.573	1.168
MCS	ANN	R^2	1.0	$1.0 \cdot 10^{-10}$	1.041	3.257
MCS	ANN	MSE	0.999	$1.5 \cdot 10^{-10}$	1.108	3.111
SSEQ	ANN	MSE	0.999	$2.3 \cdot 10^{-1}$	0.639	1.533
HSEQ	ANN	R^2	0.999	$2.0 \cdot 10^{-10}$	0.633	1.37

TABLE III

BEST FIVE SURROGATE MODELS OF THE PV PLANT ACCORDING TO THE R^2 . ANNS CONSISTENTLY DELIVER THE BEST RESULTS EVEN WHEN SORTING BY ONE OF THE OTHER SCORES.

with time stamp, radiation, and air temperature. The results are shown in Figure 5. The electrical power output prediction of the surrogate model is very accurate as long as there is actually energy generation. When there is no generation, the surrogate model predicts negative values. The module temperature seems to be captured quite accurate as well. However, the surrogate model is always one step late, but this has no visible influence on the power output.

Overall, the results for the PV plant are satisfactory in our opinion. Small flaws like the negative power output could be handled e. g. applying a $\max(0, \hat{y})$ function on the output. Therefore, no further investigations are necessary for this model.

G. Results for the Fuel Cell

The results of the surrogate generation for the fuel cell sorted according to the R^2 score are shown in Table IV. The best combinations in this case are a Monte Carlo sampling with Kriging trained by the mean average error, and a Monte Carlo sampling with Kriging trained by the R^2 score. In the

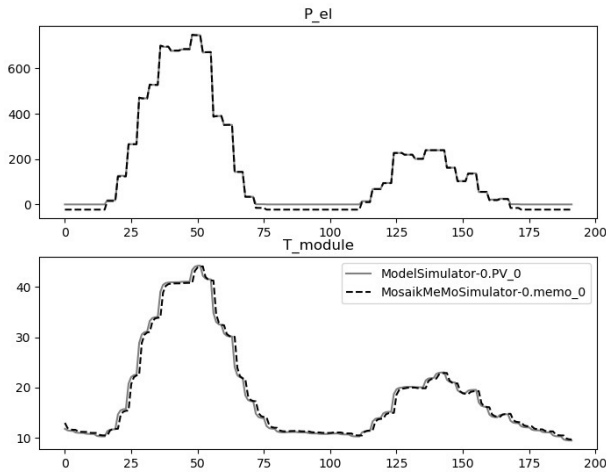


Fig. 5. Co-simulation of surrogate and original PV model comparing their electric power output. Grey line: original model, black dashed line: surrogate model.

simulation, the first configuration is used for comparison with the original model.

Sampling Strategy	Surrogate Algorithm	Train Func.	R^2 Score	HAE Score	MAE Score	MSE Score
MC	Kriging	MAE	0.997	$3.3 \cdot 10^{-10}$	1.598	30.72
MC	Kriging	R^2	0.997	$3.3 \cdot 10^{-10}$	1.598	30.72
MC	Kriging	MSE	0.996	$4.5 \cdot 10^{-10}$	2.461	43.15
LHS	Kriging	MSE	0.991	$3.5 \cdot 10^{-10}$	1.957	76.74
LHS	Kriging	R^2	0.991	$3.3 \cdot 10^{-10}$	1.957	76.74

TABLE IV

BEST FIVE SURROGATE TRAINING FOR THE FUEL CELL SIMULATION MODEL ACCORDING TO THE R^2 SCORE. THIS RANKING SHOWS THAT KRIGING IS DELIVERING THE BEST RESULTS FOR THE R^2 SCORE. .

The input schedule for electrical power is based on the standard load profile for households provided by the BDEW⁹ and for heating water we modeled a simplified schedule for the needs of a household.

The simulation result is shown by Figure 6 for the outputs: actual electrically power (P_{el}), thermal power (P_{th}), and the outgoing heating water temperature (T_{out}). The surrogate roughly follows the behavior of the original model in the output variables P_{th} and T_{out} . However, since the electrical power P_{el} is divided into discrete power stages and internally the gradient of the power is restricted so the surrogate has difficulties to reproduce the behavior especially in the transition to other power stages. The example of the fuel cell shows the difficulty of creating adequate surrogate models for models with complex internal states.

Overall, the surrogate model is satisfactory to a limited extent. If thermal power and temperature are the outputs of interest, this model performs well. For the electrical power output, however, further investigations are required.

⁹Bundesverband der Energie- und Wasserwirtschaft e.V

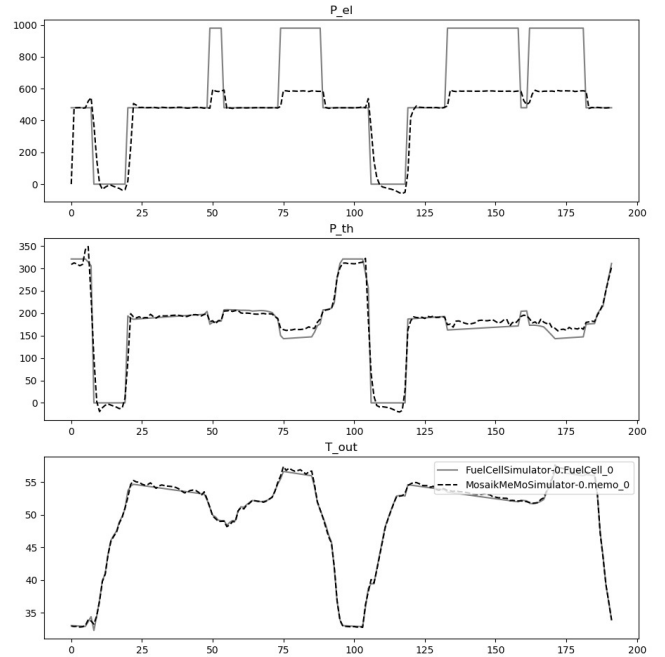


Fig. 6. Co-simulation of surrogate and original FC model. The electrical output of the model has discrete values which can be seen in the upper chart. In the middle and lower chart, the thermal outputs are shown. Grey line: original model, black dashed line: surrogate model.

VI. CONCLUSION AND FUTURE WORK

We motivated why we need surrogate models, what the challenges of the surrogate modeling process are, and which requirements a tool has to meet in order to support this process. We presented the tool MeMoBuilder that semi-automates the surrogate modeling process while testing different combinations of sampling strategies, surrogate algorithms, and training and evaluation functions to face the challenges arising when building an appropriate surrogate model. MeMoBuilder is fully compatible to the co-simulation framework mosaik and can be used on every model which can be integrated into mosaik. An integration with other co-simulation frameworks is possible by implementing the appropriate interface between MeMoBuilder and the target framework.

Further, it is possible to separate the process of sampling and the process of training. So all the requirements as described in Section II are fulfilled. We used this tool to explore the surrogate modeling process for three different simulation models and presented the results as case study.

Our major findings are a) that specific surrogate models are more suitable for concrete simulation models than other and b) the complexity of surrogate modeling can be reduced by using a tool like the MeMoBuilder. The accuracy of the presented models range from bad to good which may depend on the choice of sorting the models according to the R^2 . The MeMoBuilder provides information on which surrogate models perform well according to different criteria and gives recommendations in form of scores by different evaluation

functions.

There are still open issues which need further investigation. All provided sampling strategies and surrogate algorithms are rather generic. For some models this works quite well, for others a more specialized sampling would probably lead to better results (e. g. battery state of charge behavior). Also, the sampling designs itself are not optimized. This will be implemented in the future. Furthermore, only regression models are supported. Original models with discrete output are interpolated in the surrogate model, thus allowing values to be taken that do not exist in the original model, as shown in the fuel cell experiment. A better choice would be a classification model, but that requires training of different surrogate models for different outputs or a manual discretization of the outputs. We tried to construct the artificial neural network as a universal approximator that is generalized for many simulation models and works with a limited amount of samples. Nevertheless, there could be more suitable architectures for the individual simulation models especially with more advanced architectures like long short-term memories or convolutional neural networks.

Future studies will investigate if reducing the sample size still leads to an acceptable result since the dimensionalities of our simulation models are small. Additionally, a more advanced simulation scenario will be developed which tests the surrogate models and possible interactions with other simulation models. The next step will be integrating more specialized sampling strategies, support for classification models, and to use these surrogate models in larger scaled setups.

ACKNOWLEDGMENT

This work is supported by the European Community's Horizon 2020 Program (H2020/2014-2020) under project "ERIGrid" (Grant Agreement No. 654113). Further information is available at the corresponding website www.erigrd.eu. The conception and implementation of the MeMoBuilder tool was mainly done by Thole Klingenberg.

REFERENCES

- [1] M. Blank, T. Breithaupt, J. Bremer, A. Dammasch, S. Garske, T. Klingenberg, S. Koch, O. Lünsdorf, A. Niesse, S. Scherfke, L. Hofmann, and M. Sonnenschein, *Smart Nord Final Report*. Uni Hannover, 4 2015, pp. 21–30.
- [2] M. Blank, M. Gandor, A. Niesse, S. Scherfke, S. Lehnhoff, and M. Sonnenschein, "Regionally-specific scenarios for smart grid simulations," in *5th International Conference on Power Engineering, Energy and Electrical Drives (POWERENG2015)*. IEEE, 5 2015, pp. 250–256. [Online]. Available: <http://dx.doi.org/10.1109/PowerEng.2015.7266328>
- [3] J. P. Kleijnen, *Design and Analysis of Simulation Experiments*. Springer International Publishing, 2015. [Online]. Available: <https://doi.org/10.1007%2F978-3-319-18087-8>
- [4] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.
- [5] A. I. J. Forrester, A. Söbester, and A. Keane, *Engineering Design via Surrogate Modelling - A Practical Guide*. Wiley, 2008.
- [6] T. Simpson, J. Poplinski, P. N. Koch, and J. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, no. 2, pp. 129–150, jul 2001. [Online]. Available: <https://doi.org/10.1007%2Fpl00007198>
- [7] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene, "Multiobjective global surrogate modeling, dealing with the 5-percent problem," *Engineering with Computers*, vol. 26, no. 1, pp. 81–98, aug 2009. [Online]. Available: <https://doi.org/10.1007%2Fs00366-009-0138-1>
- [8] A. Mehmani, S. Chowdhury, C. Meinrenken, and A. Messac, "Concurrent surrogate model selection (COSMOS): optimizing model type, kernel function, and hyper-parameters," *Structural and Multidisciplinary Optimization*, vol. 57, no. 3, pp. 1093–1114, sep 2017. [Online]. Available: <https://doi.org/10.1007%2Fs00158-017-1797-y>
- [9] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. [Online]. Available: <https://doi.org/10.1016%2Fb978-0-12-804291-5.00024-6>
- [10] S. Koziel, S. Ogurtsov, and L. Leifsson, *Surrogate-Based Modeling and Optimization*. Springer New York, 2013. [Online]. Available: <https://doi.org/10.1007/978-1-4614-7551-4>
- [11] R. Pinto, R. J. Bessa, and M. A. Matos, "Surrogate model of multi-period flexibility from a home energy management system," *CoRR*, *abs/1703.08825*, 2017.
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: <http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>
- [13] K. Siebertz, D. van Bebber, and T. Hochkirchen, *Statistische Versuchsplanung - Design of Experiments (DoE)*. Springer, 2017. [Online]. Available: <https://doi.org/10.1007/978-3-662-55743-3>
- [14] C. Lemieux, *Monte carlo and quasi-monte carlo sampling*. Springer Science & Business Media, 2009. [Online]. Available: <https://doi.org/10.1007/978-0-387-78165-5>
- [15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2009.
- [16] L. Yang, S. Liu, S. Tsoka, and L. G. Papageorgiou, "Mathematical programming for piecewise linear regression analysis," *Expert systems with applications*, vol. 44, pp. 156–167, 2016.
- [17] W. Ertel, *Grundkurs Künstliche Intelligenz - Eine praxisorientierte Einführung*. Springer Vieweg, 2013. [Online]. Available: <https://doi.org/10.1007/978-3-658-13549-2>
- [18] L. Samaniego and K. Schulz, "Supervised classification of agricultural land cover using a modified k-NN technique (MNN) and landsat remote sensing imagery," *Remote Sensing*, vol. 1, no. 4, pp. 875–895, nov 2009. [Online]. Available: <https://doi.org/10.3390%2Frs1040875>
- [19] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.
- [20] C. Cui, M. Hu, J. D. Weir, and T. Wu, "A recommendation system for meta-modeling: A meta-learning based approach," *Expert Systems with Applications*, vol. 46, pp. 33–44, mar 2016. [Online]. Available: <https://doi.org/10.1016%2Fj.eswa.2015.10.021>
- [21] K. Markov and T. Matsui, "Music genre and emotion recognition using gaussian processes," *IEEE Access*, vol. 2, pp. 688–697, 2014. [Online]. Available: <https://doi.org/10.1109/ACCESS.2014.2333095>
- [22] C. Hultquist, G. Chen, and K. Zhao, "A comparison of gaussian process regression, random forests and support vector regression for burn severity assessment in diseased forests," *Remote Sensing Letters*, vol. 5, no. 8, pp. 723–732, aug 2014. [Online]. Available: <https://doi.org/10.1080%2F2150704x.2014.963733>
- [23] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225–1231, nov 1996. [Online]. Available: <https://doi.org/10.1016%2Fs0895-4356%2896%2900002-9>