Automatic Growing Design Space Exploration with Concurrent Process and Constraints Modeling (AGE-PCM)

Fabian Schneider*, Tarek Kösters*, Geritt Kampmann* and Oliver Nelles*
*Institute of Mechanics and Control Engineering – Mechatronics, University of Siegen, Siegen, Germany
(fabian2.schneider, tarek.koesters, geritt.kampmann, oliver.nelles)@uni-siegen.de

***Abstract*— Efficient exploration of constrained design spaces is a critical challenge in data-driven modeling, particularly when process knowledge is limited or when measurements are costly. Traditional Design of Experiments (DoE) approaches often separate boundary estimation and process modeling into distinct phases, leading to inefficient measurement plans. This paper introduces Automatic Growing Design Space Exploration with Concurrent Process and Constraints Modeling (AGE-PCM), a novel algorithm that integrates constraint learning and design space exploration into a single procedure. AGE-PCM begins with minimal prior knowledge, using a regularized one-class classifier to describe the feasible space incrementally and guide exploration. A two-class classifier models the boundary between feasible and infeasible space. By selectively sampling candidate points based on the classifier predictions and a space-filling metric, the resulting points are well distributed for subsequent model training. We validate the algorithm in a real-world application involving a combine harvester's threshing unit. The results demonstrate that AGE-PCM efficiently collects measurements, avoids damaging test points, and produces datasets suitable for subsequent modeling and optimization. This makes AGE-PCM an appropriate tool for constrained design space exploration and exploitation in industrial applications.**

I. Introduction

Data-driven modeling is traditionally carried out in two steps: (i) Design of Experiments (DoE) with measurement, and (ii) learning a model from the data collected in (i). Since a good distribution of the measurement data is crucial for the quality of data-driven models, this division into the two steps, DoE and model learning (with different sub-goals), is sub-optimal. Additionally, if constraints limit the design space, step (i) is often divided into sub-steps, as most DoE strategies are unsuitable for constraints [15], [17]. Considering only the DoE step, substantial difficulties arise if the process constraints are unknown. Since, for many industrial applications, limited or even no detailed process knowledge is available, it is not possible to define a design a priori and conduct process measurements within this space. Instead, first, the design space boundaries have to be identified and described. This can be done with data-driven classification models. As shown in [4]–[6], [12], [14], [16], several approaches are developed to describe the design space boundary. These range from simple polygons to support vector machines, Gaussian process classification, and regularized radial basis function (RBF) classifiers. In comparison, the latter are more advanced and better suited to describing the boundaries of complex processes. To estimate the boundary, data is measured from the process with the primary target to acquire information about the boundary. Consequently, edge-heavy point distributions result, which are unsuitable for training a data-driven process model within the constrained space. In a second step, a space-filling design is created within the explored feasible design space and measured afterward. This procedure may require more measurement points than necessary, resulting in a suboptimal point distribution. This also leads to greater effort and complexity, since different strategies must be applied and configured, increasing the complexity and the required setup time.

In [5], [12], the presented algorithms are applied to a combustion engine. In this application, the engine must be measured only once, since the process does not change over time. Nevertheless, a more efficient approach reduces the measurement time and costs. If the process changes each time, e.g., because it depends heavily on environmental variables or because each measurement point is costly, the savings are even higher. Such a process is considered the application of a combine harvester in this paper. The properties, thus the best operating point and even the feasible operating space, vary dramatically with variables such as fruit or grain humidity.

Dependent on the goal, design of experiments can be divided into two categories: first, for modeling the entire design space with high accuracy; and second, for design space exploration and optimization. In the first category, the goal is to create a design that is space-filling and uniformly distributed in the entire feasible design space. In the second case, the goal is to explore the design space non-uniformly, focusing on the most promising areas. This is often done by combining the DoE with an optimization strategy, e.g., Bayesian optimization. While the algorithm proposed in this paper focuses on the first category, several approaches have been developed for the second category. In [1], Safe Bayesian Optimization is applied on a real robotic system. The confidence intervals estimated by Gaussian process regression are used to efficiently optimize controller parameters, without risking dangerous or expensive system failures. The exploration and learning of an inverse pendulum by a reinforcement learning strategy is investigated in [2]. A strategy for save exploration of nonlinear dynamical systems is proposed in [13]. It uses a model predictive control strategy to explore the system while ensuring safety.

In this work, we propose the novel algorithm Automatic Growing Design Space Exploration with Concurrent Process and Constraints Modeling (AGE-PCM). The primary goal of

AGE-PCM is to combine all sub-steps of (i), the DoE, into one coherent step to increase the data efficiency, improve the resulting point distribution with focus on model accuracy in the entire design space, and simplify the measurement process. Besides this, it is also possible to train a desired process model over the already-known feasible space while exploring the design space. The two main components of the method are a one-class classifier (OCC) to describe the feasible region explored and a two-class classifier (TCC) to describe the design space boundary and constraints.

This paper is structured as follows: The following Sect. II presents the basics of both the OCC and TCC classifiers. Section III proposes the new method AGE-PCM. This is followed by the application to a combine harvester in Sect. IV. Section V concludes the paper with a summary and an outlook on future work.

II. Classifiers

Two different classification models form the basis of the algorithm presented in Sect III in detail. One is a One-class classifier (OCC), and the second is a two-class classifier (TCC). Both are regularized radial basis function (RBF) networks, which belong to the class of kernel-based models that utilize localized, nonlinear basis functions. First, the one-class classifier is presented, followed by the two-class classifier.

A. One-Class Classifier

The OCC proposed and applied in [6], [16] is used as a central component in AGE-PCM. Since OCCs are designed to learn a descriptive boundary around one data class, they are well-suited to describe the exploited feasible space. In addition to the basic separation into explored and unexplored space, the regularized RBF classifier also allows an estimation of the distance of a new point to the class and, thus, to the explored points. Thus, it is applied to control the propagation speed during the exploration process, which measures the distance of a new point to the existing points. A traditional RBF classifier is a set of M radial basis functions $\phi_i(\cdot)$ centered at selected points $\underline{c}_i$ in the input space $\mathcal{U}$. These functions are linearly combined by their weights w_i to model the classification function

$$g(\underline{u}) = \sum_{i}^{M} w_i \phi_i(\cdot). \tag{1}$$

As done in [6], $\phi_i(\underline{u}, \underline{c}_i, \sigma) = \exp\left(-\frac{\|\underline{u}-\underline{c}_i\|^2}{2\sigma^2}\right)$, the Gaussian function with standard deviation σ, is selected as basis function. One is centered on each of the N training points $\underline{u}_i = [u_{1,i}, u_{2,i}, \ldots, u_{d,i}]$ in the d-dimensional input space. Thus, $\underline{c}_i = \underline{u}_i$ and $M = N$. Consequently, the number of parameters w_i equals the number of data points. To avoid overfitting, which is likely to occur in this configuration, ridge regression with regularization parameter λ is used to stabilize the results. Besides the w_i estimated by least squares (LS), σ and λ are the two hyperparameters to be optimized, and the strategy is presented later.

For the LS estimation of the parameters w_i, a classification value $y_i = 1$ is defined for each data point $\underline{u}_i$. So, the LS problem can be formulated as

$$\begin{aligned} \underline{w} &= \left(\underline{X}^T \underline{X} + \lambda \underline{I}\right)^{-1} \underline{X}^T \underline{y}, \\ \text{with } X_{i,j} &= \exp\left(-\frac{(\underline{u}_i - \underline{u}_j)^T (\underline{u}_i - \underline{u}_j)}{2\sigma^2}\right), \\ \underline{w} &= [w_1, \ldots, w_M]^T \text{ and } \underline{y} = [y_1, \ldots, y_N]^T. \end{aligned} \tag{2}$$

The value of the classification function $\hat{y}_l$ for a point $\underline{u}_l$ is then calculated according to

$$\hat{\underline{y}}_l = g(\underline{u}_l) = \sum_{i}^{N} w_i \exp\left(-\frac{(\underline{u}_l - \underline{u}_i)^T (\underline{u}_l - \underline{u}_i)}{2\sigma^2}\right). \tag{3}$$

The data point $\underline{u}_l$ belongs to the class, if $\hat{\underline{y}}_l \geq y_{\text{thd}}$. Since the classification value of 1 is set for the training points, the classification threshold y_{thd} is chosen slightly smaller than the classification value, e.g., $y_{\text{thd}} = 0.99$ [6]. The value of the classification function also allows an estimation of the distance of the evaluated point to the class. The distance increases as the function decreases. This is possible because the function decreases monotonically with increasing distance from the basis functions and, thus, data points. During the exploration, new points are close to known ones, reducing the risk of severely violating the constraints.

The iterative application of the algorithm prevents the hyperparameters from being tuned with validation data. Because the algorithm starts with just one data point and increases the amount of data incrementally, one by one. Consequently, there is not enough data available. Thus, hyperparameter tuning using the training error and the leave-one-out cross-validation (LOOCV) error e_{LOO}, proposed in [5], is a suitable approach, as it does not require additional data and is computationally efficient. This is important because high computational costs would increase the measurement time during exploration and, thus, the measurement costs. A cost-saving property of the LS estimation is that the LOOCV error of the ith point can be easily and efficiently calculated with an anyway required substep of (2) according to

$$e_{\text{LOO},\,i} = \frac{y_i - \hat{y}_i}{1 - H_{i,i}}, \text{with } \underline{H} = \underline{X}\left(\underline{X}^T \underline{X} + \lambda \underline{I}\right)^{-1} \underline{X}^T, \tag{4}$$

where $\hat{y}_i$ is the value of the classification function for point $\underline{u}_i$ and $H_{i,i}$ is the ith diagonal value of the smoothing matrix $\underline{H}$. With $e_{\text{LOO},\,i}$, the classification value $\hat{y}_{\text{LOO},\,i}$ can be calculated, as if this point would not be included in the training data by

$$\hat{y}_{\text{LOO},\,i} = \hat{y}_i - e_{\text{LOO},\,i}. \tag{5}$$

A boundary as tight as possible to the given data points is desired for the AGE-PCM algorithm. The zero LOOCV error [6] enforces the robustness, since every point is classified correctly, even if it is not in the training data. The tightness of the boundary is controlled by σ, where small values lead to tighter boundaries. Consequently, the hyperparameter

optimization problem can be formulated as

$$\begin{aligned} \min_{\sigma,\lambda} \quad & \sigma \\ \text{s.t.} \quad & \hat{y}_i \geq 1, \quad \forall i = 1, \ldots, N \\ & \hat{y}_{\text{LOO},i} \geq 1, \quad \forall i = 1, \ldots, N. \end{aligned} \tag{6}$$

The first constraint ensures that all training points are classified as feasible, and the second constraint ensures that the LOOCV error is zero.

B. Two-Class Classifier

In contrast to the unsupervised one-class classification, a two-class classification is a supervised task. The goal is to distinguish between two classes. In AGE-PCM, the objective is to distinguish between feasible and infeasible points.
The chosen TCC is a modification of the previously presented regularized RBF OCC. In the two-class case, each class is assigned a value. The classification value $y_{i,\text{ feasible}} = 1$ is assigned to all feasible points, and $y_{i,\text{ infeasible}} = -1$ to the infeasible ones. The classification threshold that defines to which class a point belongs is set to $y_{\text{thd, TCC}} = 0$, e.g., points with positive classification function value $g(\underline{u})$, see (1), are classified as feasible. In this way, the problem is defined so that the weights w_i can still be estimated with LS, and the LOOCV error can still be calculated according to (4). However, the hyperparameter optimization needs to be modified as well. Instead of minimizing σ, in the two-class case, the goal is to maximize σ. Unlike the OCC case, where a tight boundary around the points is desired, the TCC describes the border between the two classes. This boundary should be as smooth as possible, which is achieved by a large σ. This also reduces the boundary's flexibility. The robustness properties of the OCC remain desirable. Thus, the training error and LOOCV error have to be zero. For this, two sets are defined, where $\mathcal{F}$ contains all feasible points, and $\mathcal{I}$ contains the infeasible points of the training data. This leads to the following formulation:

$$\begin{aligned} \max_{\sigma,\lambda} \quad & \sigma \\ \text{s.t.} \quad & \sigma \in [\sigma_{\text{TCC,min}}, \sigma_{\text{TCC,max}}] \\ & \sum_{i\in\mathcal{F}} H\left(-\hat{y}_i(\sigma,\lambda)\right) + \sum_{i\in\mathcal{I}} H\left(\hat{y}_i(\sigma,\lambda)\right) = 0 \\ & \sum_{i\in\mathcal{F}} H\left(-\hat{y}_{\text{LOO},i}(\sigma,\lambda)\right) + \sum_{i\in\mathcal{I}} H\left(\hat{y}_{\text{LOO},i}(\sigma,\lambda)\right) = 0 \end{aligned} \tag{7}$$

where $H(\cdot)$ is the binary step function according to

$$H(x) = \begin{cases} 1, & \text{if } x > y_{\text{Lim, TCC}} \\ 0, & \text{else} \end{cases}. \tag{8}$$

The classification value $\hat{y}_i(\sigma,\lambda)$ of the i-th training point is obtained according to (3). In the LOOCV case, the classification value $\hat{y}_{\text{LOO},i}(\sigma,\lambda)$ is calculated according to (5). Consequently, the second constraint ensures zero training error, and the third constraint ensures zero LOOCV error.
The demanded zero LOOCV error restricted the smoothness of the function and thus serves as an upper bound for σ depending on the complexity of the classification problem. If all hyperparameter combinations violate the constraints, the training error and zero LOOCV error are set to the smallest possible value. This behavior is visualized in Fig. 1. For this, a two-dimensional test case is created. The design space is defined by

$$\begin{aligned} \mathcal{D} &= \left\{\underline{u} = [u_1, u_2]^T \in [0,1]^2\right\}, \\ \text{with } \mathcal{D}_{\text{feasible}} &= \left\{\underline{u} \in \mathcal{D} \,\middle|\, u_1 u_2 < \frac{1}{6}\right\}. \end{aligned} \tag{9}$$

This test process is also used in Sect. III to explain and visualize AGE-PCM.
This optimization strategy ensures an easy-to-implement and application-independent selection of hyperparameters without requiring extensive prior knowledge.

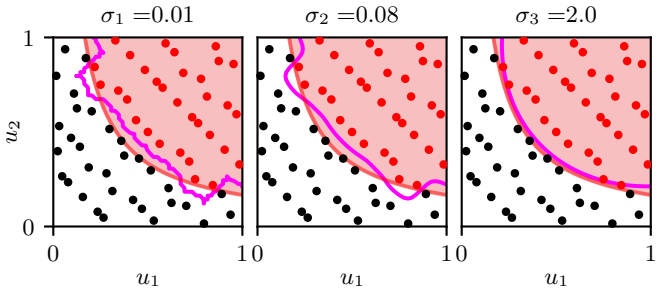

Fig. 1. Estimated classification boundary for three different σ. The red curve describes the boundary of the feasible space, and the infeasible space is colored light red. The training points are created by a maximin Latin hypercube [3]. The feasible ones are marked in black, and the infeasible ones are marked in red. Left plot: small σ leading to a rough boundary with zero training and LOOCV error. Middle plot: σ optimized according (7). The boundary is smoother than the left one, and the errors are still zero. Right plot: Large σ, smooth boundary, but training and LOOCV error are larger than zero.

III. Algorithm

In this section, the AGE-PCM algorithm is explained. The algorithm starts at a known feasible operating point. From this point, the algorithm explores the design space. The resulting point distribution in the feasible space should be as space-filling as possible to ensure good design quality for later tasks such as modeling. At the same time, the constraint violations should be kept to a minimum to ensure safe operation. Infeasible points deep in infeasible territory should be avoided.
The algorithm is based on candidate points to reduce computational costs and discretize the design space. The simplest way to do this is a grid over the entire design space. The d-dimensional design space is divided into an equally spaced grid with L levels in each dimension. Thus, a grid with L^d points results. Due to the curse of dimensionality, the grid-based approach becomes inefficient in high-dimensional applications. The number of candidate points increases exponentially with the number of dimensions. In this case, extended variants based on Sobol sequences or optimized Latin hypercube sampling can be used to generate candidate points more efficiently. In the following, due to the low-dimensional examples, the grid-based approach is used. The

algorithm can test only these finite number of points, which are called candidate points $\mathcal{C}$.

Besides L, the algorithm has the following hyperparameters: N_{Lim}, $y_{\text{expl, OCC}}$, σ_{OCC}, λ_{OCC}, σ_{TCC} and λ_{TCC}. The first is the maximum number of points N_{Lim} that can be measured. This is necessary to limit the computational effort. The second is the classification value of the one-class classifier $y_{\text{expl, OCC}}$. This controls the algorithm's exploration progress. Besides this, for both the OCC and TCC, the hyperparameters σ and λ are required. Those control the smoothness of the classifier and are determined according to the hyperparameter optimization strategies according to (6) and (7). The TCC requires also the interval $[\sigma_{\text{TCC,min}}, \sigma_{\text{TCC,max}}]$. The algorithm can be applied if these parameters are set and a feasible initial point is selected. The algorithm is shown in Algorithm 1. Figure 2 shows AGE-PCM applied to the test case defined in (9). The algorithm starts with a known feasible point

Algorithm 1 Design Space Exploration with AGE-PCM

Initialize sets: $\mathcal{F} \leftarrow \emptyset$ (feasible), $\mathcal{I} \leftarrow \emptyset$ (infeasible)
Generate candidate points $\mathcal{C}$ by a d-dimensional grid with M levels in each dimension over the design space
Start at a known feasible point $\underline{u}_0 \in \mathcal{C}$ (safe point)
$\mathcal{F} \leftarrow \mathcal{F} \cup \{\underline{u}_0\}$
Remove $\underline{u}_0$ from $\mathcal{C}$
Train one-class classifier $g_{OCC}(\cdot)$ on $\mathcal{F}$
while termination condition not met **do**
    $\mathcal{C}^* \leftarrow$ candidates from $\mathcal{C}$
    $\mathcal{C}^* \leftarrow \mathcal{C}^* \setminus \underline{u}_i$ if $g_{\text{OCC}}(\underline{u}_i) \notin I_{\text{expl}}$, $\forall\, \underline{u}_i \in \mathcal{C}^*$
    if $\mathcal{I} \neq \emptyset$ **then**
        $\mathcal{C}^* \leftarrow \mathcal{C}^* \setminus \underline{u}_i$ if $g_{\text{TCC}}(\underline{u}_i) < y_{\text{thd, TCC}}$, $\forall\, \underline{u}_i \in \mathcal{C}^*$
    end if
    if $\mathcal{C}^* = \emptyset$ **then**
        break
    end if
    Compute $\phi_p(\mathcal{F} \cup \{\underline{u}_i\})$ for each $\underline{u}_i \in \mathcal{C}^*$
    Select $\underline{u}^* \leftarrow \arg\max \phi_p(\underline{u}_i)$
    Measure if $\underline{u}^*$ is feasible in the real process
    if $\underline{u}^*$ is feasible (in ground truth) **then**
        If required, measure the process output
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\underline{u}^*\}$
        Retrain $g_{\text{OCC}}(\cdot)$ on updated $\mathcal{F}$
    else
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{\underline{u}^*\}$
    end if
    if $\mathcal{I} \neq \emptyset$ **then**
        Retrain $g_{\text{TCC}}(\cdot)$ on updated $\mathcal{F}$ and $\mathcal{I}$
    end if
    Remove $\underline{u}^*$ from $\mathcal{C}$
end while

$\underline{u}_0 = [0.35, 0.35]^T$ and initializes the set of feasible points $\mathcal{F}$ with this point. The algorithm then enters a loop where it iteratively selects and tests candidate points. In each iteration, the algorithm first filters out candidate points outside the exploration interval defined by the one-class classifier. Then, it filters out candidate points that are *predicted* to be infeasible by the two-class classifier. If no candidate points remain, the algorithm terminates prematurely. Otherwise, a quality metric measuring the space-fillingness of the feasible points and each candidate point is calculated. The point optimizing the metric is selected as $\underline{u}^*$. The selected point is then *measured* in the real process. It is added to the set of feasible points $\mathcal{F}$ if it is feasible. Otherwise, it is added to the set of infeasible points $\mathcal{I}$. The algorithm then retrains both classifiers based on the updated sets $\mathcal{F}$ and $\mathcal{I}$. This process continues until a termination condition, here the maximum number of points N_{Lim}, is met.

At this point, the exploration interval and the quality metric need to be explained. Since only small violations of the constraints are allowed, the exploration progress in each iteration needs to be controlled. This is done by the OCC and by controlling the exploration interval $I_{\text{expl}} := [y_{\text{expl, OCC}}, y_{\text{lim, OCC}}]$. Therefore, a point $\underline{u}_i$ with a classification value $y_{OCC,i} = g_{\text{OCC}}(\underline{u}_i)$ smaller than $y_{\text{lim, OCC}}$ is outside the known class and thus outside the already explored space. The value of $y_{\text{expl, OCC}}$ on the other side is used to restrict the distance of the point to the explored space. If a point $\underline{u}_i$ has a classification value $y_{\text{OCC, i}}$ smaller than $y_{\text{expl, OCC}}$, it is considered to be too far away from the already explored space and thus is not allowed to be measured. This procedure allows the algorithm to explore the design space in a controlled manner. The area between the blue lines shows this exploration interval in Fig. 2. Here, the dark blue line is the lower bound $y_{\text{expl, OCC}}$ and the light blue line is the hull around the cloud of feasible points and thus corresponds to the upper bound $y_{\text{lim, OCC}}$.

Besides the controlled exploration of the design space, the second central task of the algorithm is to find a good distribution of the points, more precisely, a space-filling distribution, since such designs improve the quality of later trained nonlinear models [11]. This is encouraged by the quality metric $q(\mathcal{F} \cup \{\underline{u}_i\})$, on which the algorithm selects the next test point $\underline{u}^*$. In general, every space-filling metric can be deployed. However, we propose to utilize the ϕ_p criterion [10], which is defined as

$$\phi_p(\mathbb{U}) = \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|\underline{u}_j - \underline{u}_i\|^{-p} \right\}^{1/p}, \qquad (10)$$

with $\mathbb{U} = \{\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_N\}$ being the set of all points. The parameter p is a hyperparameter and controls the smoothness of the function. The larger p, the more the smallest nearest-neighbor distance of the design is emphasized. For $p \to \infty$, it approaches the reciprocal of the minimal nearest-neighbor distance. According to recommendation in [10] $p = 15 \ldots 50$, $p = 15$ is chosen in this work.

The ϕ_p criterion is selected because it is a distance-based, computationally efficient metric for small data sets. In addition, designs optimized by the ϕ_p criterion are well suited for nonlinear data-driven models. No reference distribution is needed for distribution-based metrics, e.g., the Kullback-Leibler divergence [7]. Since the feasible space is unknown,

the reference distribution is also unknown. In summary, the

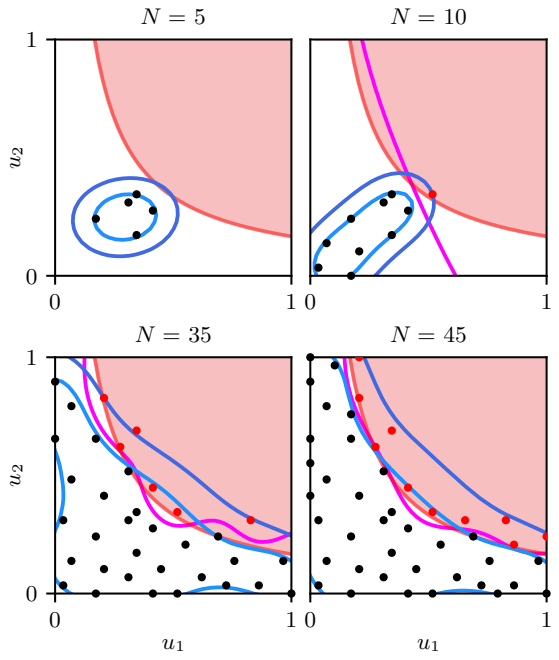

Fig. 2. AGE-PCM after $N = 5$, $N = 10$, $N = 35$ and $N = 45$ points applied to a test case. The blue lines describe the exploration interval calculated by the OCC. The magenta line describes the classification boundary of the TCC. The red area defines the true infeasible space according to (9).

ϕ_p criterion drives the points apart, and the exploration interval keeps the new points close to the old. This procedure can achieve a good trade-off between exploration and save-operation.

Figure 2 shows the status of AGE-PCM after several iterations. Here, the following hyperparameters are used: $M = 30$, $N_{\text{Lim}} = 45$, and $y_{\text{expl, OCC}} = 0.9$. The classification thresholds are set as $y_{\text{lim, OCC}} = 0.99$, $y_{\text{lim, TCC}} = 0$. The hyperparameters σ_{OCC}, λ_{OCC}, σ_{TCC} are optimized. Therefore, σ_{TCC} is limited to $\sigma_{\text{TCC,min}} = 0.05$ and $\sigma_{\text{TCC,max}} = 0.3$. The first point is selected as the safe point, which is known to be feasible. The first five points are then measured, and the algorithm is visualized after each iteration.

The first five points are all feasible. Consequently, the TCC could not be trained at this point. The plots are generated after each iteration. Thus, the OCC and TCC are shown retrained for all visualized points. One point is infeasible for $N = 10$ and violates the constraint slightly. The TCC can now be trained. As expected, with only one infeasible point, the TCC is inaccurate. The points are neither too close to each other nor too far apart. After $N = 35$ points, the feasible space is almost completely explored. Only the upper left corner is not yet covered. The feasible points are distributed equally over the explored space. Also, the TCC's accuracy improved. As intended, the infeasible points are close to the boundary, and the infeasible points are only a small fraction of all points measured. Since the exploration is almost complete, the next ten points are mainly used to improve the TCC. Adding more points near the boundary improves the quality of the constraint model. The algorithm terminates at this point because the point limit, $N_{\text{Lim}} = 45$, is reached. The feasible space has also been approximated well by the OCC. Visually, the resulting feasible design has appropriate space-filling properties. The points are well distributed and close to the boundaries.

The following remarks address important aspects of the algorithm. Regarding initialization, it is generally important to start with a feasible point; otherwise, a strategy to find one must be implemented. Both OCC and TCC require at least one feasible point for training. Without a known feasible point, the algorithms cannot be applied. However, while a starting point in the center of the feasible space is preferred, the algorithm is robust against the point selection as demonstrated in Fig. 3.

A second aspect is the application of the algorithm to higher-

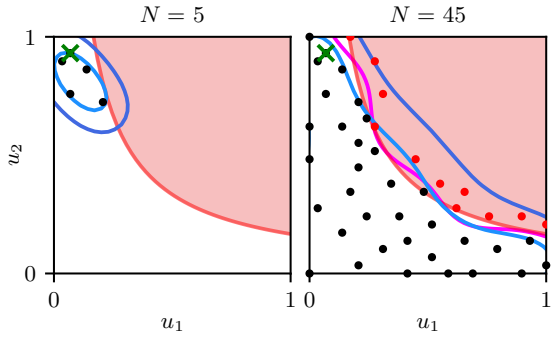

Fig. 3. AGE-PCM with a different initial point after $N = 5$ and $N = 45$ points applied to a test case. The green cross marks the initial point.

dimensional problems. Conceptually, the algorithm can be applied to more than three or four dimensions. However, some aspects need to be considered. As mentioned before, the grid-based approach for candidate point generation becomes inefficient in higher dimensions. Thus, more efficient sampling strategies are required. Since the classifier training and hyperparameter optimization are computationally efficient, computational costs are less of a concern. The distance-based quality metric ϕ_p is also applicable in more than four dimensions.

Additionally, the algorithm is designed to be applied to real processes. Thus, measurement noise can be expected, leading to misclassification of points. In its current form, the algorithm is not designed to handle noise explicitly. However, the hyperparameter optimization strategy allows a small amount of misclassification, thereby reducing the influence of noise. Nevertheless, extending the algorithm to estimate the probability of misclassification, perhaps incorporating it into the selection of new points, or even removing this point, is a promising direction for future work. Regarding the TCC, the roughness of the boundary depends on the hyperparameter σ_{TCC}. Here, further research is required to select a suitable σ_{TCC} automatically without the need for a user-defined interval. This problem becomes harder if

the data contains two close feasible and infeasible points. Then, it is difficult to find a smooth boundary with zero classification error.

Despite this potential for improvement, by combining the OCC and TCC, the proposed algorithm enables safe, controlled exploration of the feasible design space. The LOOCV hyperparameter optimization enables the algorithm to automatically adapt to the process's smoothness and constraints without requiring any prior knowledge. Therefore, AGE-PCM can be easily applied to various processes without the need for complex hyperparameter tuning.

IV. Application

In this section, AGE-PCM is applied to a real-world application to explore the suitable operating space of the threshing unit of a combine harvester.

The threshing unit is a central component of a combine harvester and typically contains one or more threshing drums, and the threshing concave. Their task is to rip out the grains of the ears and first separate the grains and materials other than the grains. Subsequently, this separation is intensified within the upcoming sections inside the combine harvester. For more information about threshing units, refer to [8], [9]. In this work, the threshing unit is investigated solely.

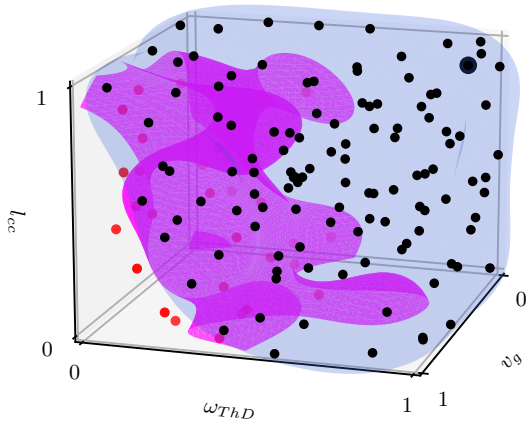

Fig. 4. Result of AGE-PCM applied in a harvesting test with $N = 150$ different operation points. The threshing drum rotational speed ω_{ThD}, the ground speed v_g and the concave clearance l_{cc} are varied. Black dots: feasible operation points. Red dots: Infeasible operation points. Red surface: Modeled process boundary of the threshing drum by TCC. Blue surface: Hull around the feasible points according to OCC.

Here, the harvested material flow has to pass between the threshing drum(s) and the concave. The combine harvester is equipped with adjustment options inside the threshing unit to adapt to the current material conditions. The first option is the concave clearance l_{cc}, which the operator can set. The second adjustable option is the threshing drum rotational speed ω_{ThD}. Meanwhile, the third option is the ground speed v_g, which controls the mass flow through the threshing unit. Specific to the harvested material, it is known that different settings are optimal for realizing a good threshing and separation result. A well-performing range of settings can vary dramatically for various fruits and within the same fruit depending on the material conditions, e.g., straw and grain humidity. Data-driven models and ensuing mathematical optimization are used to figure out well-performing adjustments. These models need to be trained with data measured under the actual conditions. Thus, space-filling data is required to ensure a good model quality, resulting in good optimization results. Data efficiency is essential to reduce the measurement time and, thus, the operation time with inefficient settings. Besides the optimal settings, it is

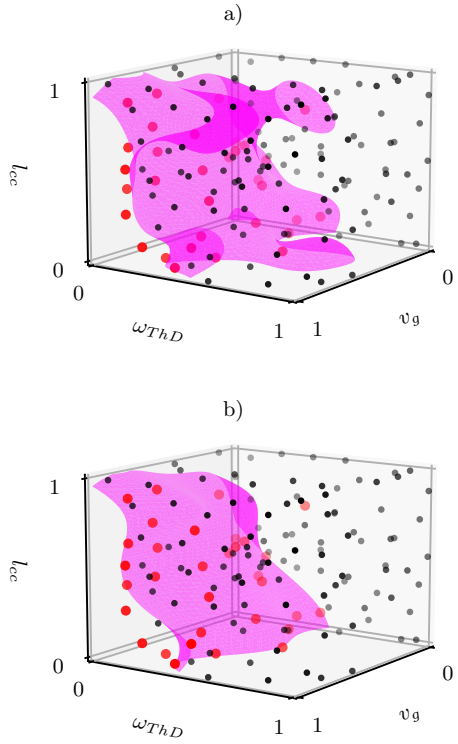

Fig. 5. Comparision of two different TCCs. a) $\sigma = 0.27$ optimized according (7). b) $\sigma = 0.5$ manually set. The red surface describes the modeled boundary according to the TCC. The black dots are the feasible points, and the red ones are the infeasible points.

also known that bad settings can lead to interruptions or in severe cases to breakdowns of the process as a consequence of blocking the threshing drum. Clearing the blockage requires hard manual labor and costs threshing time. Hence, while gathering data for modeling, critical settings must be avoided as much as possible. However, bad settings typically do not directly lead to break-downs of the threshing unit; they notify themselves acoustically and are well recognizable to the combine operator. Therefore, this work uses the operator's feedback, derived from personal impression, to indicate infeasible settings. The stated algorithm investigates the desired space as equally as possible by avoiding critical

settings from being set unnecessarily, often in an efficient way. Thus, AGE-PCM is well-suited for this application. The final result of AGE-PCM is shown in Fig. 4 with the hyperparameters as follows: $M = 25$, $N_{\text{Lim}} = 150$, $y_{\text{expl, OCC}} = 0.9$. The results show that also, in the three-dimensional case, the algorithm can explore and model the feasible space. The resulting feasible points are suited for subsequent modeling and optimization tasks. Critical combinations of small l_{cc} and small ω_{ThD} are entirely avoided.

The TCC used during exploration is shown in Fig. 5 a). The rough boundary might be unsuitable for the subsequent optimization, which seeks well-performing settings for the threshing unit. Additionally, a single infeasible point is observed surrounded by feasible points. This point might be an outlier or be misclassified due to noise. Both problems can be solved by restricting the flexibility of the TCC by increasing the hyperparameter σ_{TCC}, see Fig. 5 b). The TCC is much smoother, and the outlier point is not classified as infeasible. However, σ_{TCC} is not optimized and thus leads to a larger LOOCV error and a training error (outlier point).

V. Conclusion

This paper presents AGE-PCM (Automatic Growing Design Space Exploration with Concurrent Process and Constraints Modeling), a novel algorithm for data-efficient and safe exploration of constraint design spaces. The proposed method systematically explores an unknown feasible region while minimizing constraint violations and ensuring efficient data utilization.

The primary contribution of this work is the integration of one-class and two-class classifiers into a unified framework that jointly performs design space exploration, constraint modeling, and data generation. By combining multiple substeps of design of experiments into a single process, AGE-PCM improves data efficiency compared to traditional sequential approaches. By selecting points based on the ϕ_p criterion, the algorithm ensures that measured points are well distributed throughout the feasible space, thereby improving the quality of subsequently trained nonlinear models. Leave-one-out cross-validation (LOOCV)-based hyperparameter optimization reduces the need for manual tuning or expert knowledge, enabling application-independent deployment with minimal prior knowledge. The successful application to a real combine harvester threshing unit demonstrates the practical value of AGE-PCM in a complex, real-world application.

The algorithm should be enhanced at this point to handle high-dimensional design spaces more effectively as the grid-based candidate point generation becomes inefficient. Additionally, the two-class classifier's boundary can become rough depending on the point distribution and the hyperparameter optimization. This can lead to unsuitable boundaries for subsequent optimization tasks. Besides this, potential misclassification of points due to measurement noise or errors is not explicitly handled. Future work should address these limitations by developing alternative candidate-point generation strategies for high-dimensional spaces, improving classifier techniques, and addressing misclassification handling.

Overall, the algorithm achieves an effective trade-off among exploration efficiency, safety, and model quality, making it well-suited for automated, data-efficient process exploration.

References

[1] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, Stockholm, Sweden, May 2016. IEEE.

[2] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 908–919, Red Hook, NY, USA, 2017. Curran Associates Inc.

[3] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann, and Oliver Nelles. Extended deterministic local search algorithm for maximin latin hypercube designs. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 375–382, 2015.

[4] Benjamin Hartmann, P. Heuser, E. Kloppenburg, and R. Diener. Online-methods for engine test bed measurements considering engine limits. In Michael Bargende, Hans-Christian Reuss, and Jochen Wiedemann, editors, *16. Internationales Stuttgarter Symposium*, pages 1251–1264, Wiesbaden, 2016. Springer Fachmedien Wiesbaden.

[5] Geritt Kampmann, Nataša Kieft, and Oliver Nelles. Support Vector Machines for Design Space Exploration. In *World Congress on Engineering and Computer Science*, pages 1116–1121, Berkeley, CA, USA, October 2012.

[6] Geritt Kampmann and Oliver Nelles. One-Class LS-SVM with Zero Leave-One-Out Error. In *IEEE Symposium Series On Computational Intelligence*, pages 1–6, Orlando, Florida, USA, December 2014. IEEE.

[7] Solomon Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[8] Tarek Kösters, Tim O. Heinz, and Oliver Nelles. Optimization based excitation signal design tailored to application specific requirements. *IFAC-PapersOnLine*, 55(37):451–456, 2022. 2nd Modeling, Estimation and Control Conference MECC 2022.

[9] Tarek Kösters, Christopher Illg, and Oliver Nelles. Adaptive nonlinear system identification of separation unit for raw materials. *Control Engineering Practice*, 162:106370, 2025.

[10] Max D. Morris and Toby J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

[11] Iván Negrin, Moacir Kripka, and Víctor Yepes. Metamodel-assisted design optimization in the field of structural engineering: A literature review. *Structures*, 52:609–631, June 2023.

[12] H. Oyama, M. Yamakita, K. Sata, and A. Ohata. Identification of static boundary model based on gaussian process classification. *IFAC-PapersOnLine*, 49(11):787–792, 2016. 8th IFAC Symposium on Advances in Automotive Control AAC 2016.

[13] Manish Prajapat, Johannes Köhler, Matteo Turchetta, Andreas Krause, and Melanie N. Zeilinger. Safe Guaranteed Exploration for Nonlinear Systems. *IEEE Transactions on Automatic Control*, 70(8):5333–5348, August 2025.

[14] Adrian Prochaska, Julien Pillas, Klaus Lüpkes, Yagiz Dursun, Reiner Pätzold, and Bernard Bäker. Approach for online design of experiments with additional constraint modeling. In Johannes Liebl, editor, *Der Antrieb von morgen 2020*, pages 172–183, Wiesbaden, 2021. Springer Fachmedien Wiesbaden.

[15] Fabian Schneider, Ralph J. Hellmig, and Oliver Nelles. Latin hypercubes for constrained design of experiments for data-driven models. *at - Automatisierungstechnik*, 71(10):820–832, 2023.

[16] Volker Smits, Max Schüssler, Geritt Kampmann, Christopher Illg, Tim Decker, and Oliver Nelles. Excitation Signal Design and Modeling Benchmark of NOx Emissions of a Diesel Engine. Trieste, Italy, 2022.

[17] Zhixiang Wang, Dapeng Zhang, Yongjun Lei, Zeping Wu, Jie Wang, Xing OuYang, and Jun Wang. Constrained space-filling and non-collapsing sequential design of experiments and its application for the lightweight design of cylindrical stiffened shells. *Structural and Multidisciplinary Optimization*, 64(6):3265–3286, September 2021.