A SAMPLING-BASED OPTIMIZATION ALGORITHM FOR SOLUTION SPACES WITH PAIR-WISE COUPLED DESIGN VARIABLES

HELMUT HARBRECHT, DENNIS TRÖNDLE, AND MARKUS ZIMMERMANN

Abstract. Solution spaces are sets of good designs that satisfy all design goals. They serve as target regions for robust and independent component development in a distributed design process. So-called solution boxes provide best decoupling, however, they are often small and therefore impractical. This article proposes an algorithm that computes two-dimensional permissible regions for pairs of design variables that are substantially larger than solution boxes. This is accomplished by modifying the existing sampling-based optimization algorithm for boxes and extending it by box-rotation.

1. Introduction

Uncertainty is present in many engineering problems, especially in the early development phase. The uncertainty relevant for this article emerges in distributed design processes where system designers, i.e., engineers that are responsible for the overall system performance, specify design goals for component designers, i.e., engineers that are responsible for component performance. From a system designer's perspective, component performances are the design variables, denoted as $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. They are uncertain in an epistemic sense, since, first, they are not known exactly in an early development stage, and, second, this uncertainty will be removed during the course of the design process. The source of this uncertainty lies in the nature of a distributed development process. When a system designer requests a particular performance of a component designer he/she cannot be certain about the result. The request may be technically impossible to realize, too expensive or be in conflict with other requirements. Eventually, when the design is finalized, the uncertainty is removed by confirmation that a component design actually can be realized. The approach presented here enables design in presence of this kind of epistemic uncertainty.

If there were no uncertainty, the goal of computation would be to find the optimal design variables with respect to an objective function f out of a space of admissible designs Ω_{ds}. This would lead to solving an optimization problem

$$f(\boldsymbol{x}) \to \min_{\boldsymbol{x} \in \Omega_{\mathrm{ds}}} . \tag{1.1}$$

Popular methods to solve this kind of *black-box optimization problem* are the *Nelder-Mead method* and genetic algorithms such as *differential evolution* (see [1, 16, 20]). However, when uncertainties are present in the problem, it is possible to deal with them by increasing the robustness of the solution. This could be done by *robust design optimization*, *reliability-based design optimization* or *sensitivity analysis*, see [2, 3, 4, 5, 6, 21, 22] for example. These methods require certain assumptions on the problem under consideration. For robust design optimization and reliability-based design optimization, the variability of the design variables has to be known, which may be expressed by probability density functions. Unfortunately, this data is typically not available for the uncertainty related to distributed development processes. In this case, an alternative method as proposed in this article may be applied.

In addition to treating uncertainty appropriately, the following challenges are to be overcome:

- The design variables $x_1, \dots, x_d$ are coupled with each other, i.e., they simultaneously affect the overall system performance.
- The evaluation of f is expensive. Therefore, it is mandatory to keep the number of function evaluations small.
- f is a black-box function. It is possibly noisy and no information about the gradient is available. Hence, classical optimization techniques cannot be applied.
- There are a large number of design variables, so Ω_{ds} is high-dimensional.

One approach to deal with these challenges is *set-based design*, cf. [18, 15, 17]. Before committing to a design, the design teams identify sets of feasible designs, e.g., expressed as regions of permissible design variable intervals. Design teams restrict themselves to work only with design variables from these sets.

This article extends a computational method to optimize high-dimensional subsets of feasible designs, so-called *solution spaces*, proposed in [23]. Synonym for solution spaces are *permissible design spaces* or *feasible design areas*, cf. [7, 10]. On a solution space, the objective function f assumes only subcritical output values, i.e., they are *good designs*.

Solution spaces are used to treat uncertainty in a particular sense. They are not used to predict what deviation from the intended system performance will occur, but rather what tolerances for intended component performances are required. Tolerances for component performances may be optimized such that the number of permissible designs, i.e., the overall tolerance is maximized. This approach assumes a top-down view for systems design rather than a bottom-up view for uncertainty propagation. It is particularly useful in situations where data about uncertainty, like distribution functions, are

not available, since it is not required for this approach. Epistemic uncertainty in early development stages lacks this information and can therefore be treated with solution spaces. The same is true for aleatory uncertainty, however, in early design stages of distributed development processes, epistemic uncertainty tend to be more relevant, see [24].

Definition 1.1. A design $\boldsymbol{x}$ is called a *good design* or a *good design point* if $f(\boldsymbol{x}) \leq c$ and *bad design* or *bad design point* if $f(\boldsymbol{x}) > c$ for a critical value $c \in \mathbb{R}$ which is given by the problem. Additionally, the set of all good designs is defined as the complete solution space

$$\Omega_c := \{\boldsymbol{x} \in \Omega_{\text{ds}} : f(\boldsymbol{x}) \leq c\}.$$

The key idea of [23] has been to express the solution space sought by intervals for each input design variable, thus representing a high-dimensional, axis-parallel box Ω_{box}:

$$\Omega_{\text{box}} := \prod_{i=1}^{d} [a_i, b_i] \subset \Omega_{\text{ds}}.$$

A design will always evaluate as good, as long as the values of its design variables remain within their respective intervals. By specifying target intervals that do not depend on the choice of interacting design variables, design variables are said to be uncoupled, enabling the independent development of the involved components. As the size of the intervals and thus the volume of the box are to be maximized, the following semi-infinite optimization problem can be formulated:

Maximize the volume

$$\mu(\Omega_{\text{box}}) \to \max_{\Omega_{\text{box}} \subset \Omega_{\text{ds}}}$$

over all axis-parallel boxes $\Omega_{\text{box}} \subset \Omega_{\text{ds}}$ subject to

$$f(\boldsymbol{x}) \leq c \text{ for all } \boldsymbol{x} \in \Omega_{\text{box}}.$$

This problem can be solved by applying the algorithm proposed in [23] and studied in [13], which will be referred to as *(sampling-based) box optimization*. The algorithm starts by constructing a small box Ω_{box} somewhere in Ω_{ds}. Then, during the so-called *exploration phase*, design points are sampled in Ω_{box}. Bad designs are removed by appropriately modifying the box boundaries (*trimming*) and extending it again (*growing*). Applying this procedure, the box will move towards a large area of good design space in Ω_{ds}. Finally, in the *consolidation phase*, the box is trimmed until no more bad design points are detected by sampling.

The box optimization is well established in vehicle design, see [8, 12, 13, 23] for example. In applications, however, often some design variable intervals are too small for practical use. For example, if Ω_c takes the form of a diagonal strip in two dimensions (see Fig. 1), an axis-parallel box will be small in relation to the whole Ω_c. In this article, the original problem statement is

therefore modified to allow two-dimensional box-rotations for specific design variable pairings. On the one hand, this introduces coupling between these pairs (note that pairs remain uncoupled from each other). On the other hand, this increases the solution space considerably, especially if strongly correlated design variables are taken as pairs. It is of note that the coupling of pairs of design variables has been studied in [9] in the case when f is a linear function. The algorithm proposed there finds very large solution spaces by using an *interior-point algorithm* (compare [19]).

The rest of this article is structured as follows. In Section 2, the concepts of 2D-maps and the box-rotation are explained. Then, in Section 3, the box optimization is extended in order to handle the box-rotation. The numerical experiments and the comparison of the box-rotation algorithm to the box optimization are described in Section 4. Finally, the conclusion is drawn in Section 5.

2. Box-rotations for 2D-maps

In order to allow for box-rotations, the concept of *2D-maps* is utilized. They have been introduced as *2D-spaces* in [9].

Definition 2.1. The *2D-map* $\Omega_{i,j}$ is defined as

$$\Omega_{i,j} := \left\{ \boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{y} = \pi_{i,j}(\boldsymbol{x}),\, \boldsymbol{x} \in \Omega_{\mathrm{ds}} \right\},$$

where $\pi_{i,j}$ is the projection $(x_1, \ldots, x_d) \mapsto (x_i, x_j)$ with $1 \leq i, j \leq d$. That is, the 2D-map $\Omega_{i,j}$ is the projection of Ω_{ds} onto the dimensions i and j.

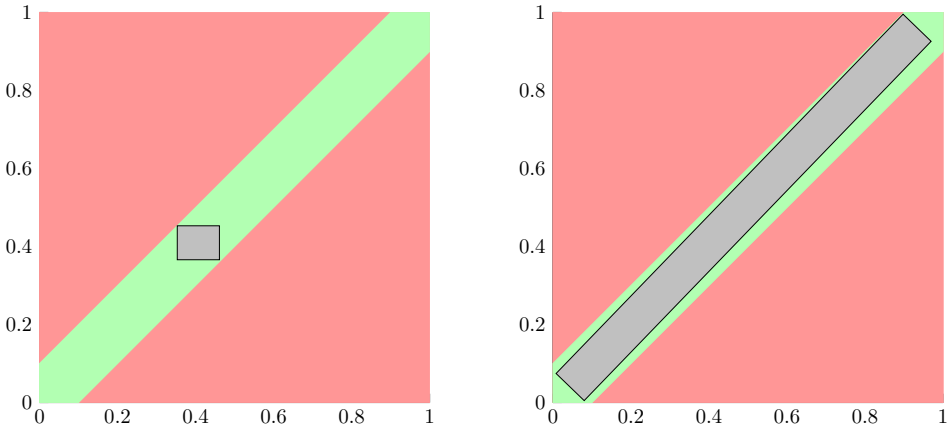

Figure 1. An axis-parallel box (left) and a rotated box (right) as solution spaces. Good and bad design regions are green and red, respectively.

The box-rotation algorithm extends the box optimization by the following: Two design variables x_i and x_j are paired and associated with the 2D-map

$\Omega_{i,j}$. The projection $\pi_{i,j}(\Omega_{\text{box}})$ in $\Omega_{i,j}$ is rotated such that it is no longer axis-parallel in these two dimensions. The design variables x_i and x_j are thereby coupled again. However, this trade-off is acceptable since the coupling does not include the other design variables. This means that changes can be made to the design variable x_i, for example, and only x_i and x_j have to be checked whether they still lie within the rotated box $\pi_{i,j}(\Omega_{\text{box}})$. The other design variables are not affected, which saves time and resources during the design process.

A meaningful choice of design variables to be coupled in one 2D-map may be inferred from the design problem. It may make sense, e.g., to couple design variables associated with one component while keeping them uncoupled from those associated with another component. Each design variable is paired with at most one other design variable, such that each design variable is associated with at most one 2D-map. Some design variables may not need to be paired with any other design variables, usually because they are expected to have enough available design space. These design variables are assigned to intervals, as in the box optimization. Therefore, the box Ω_{box} is the product of one-dimensional intervals I_k and two-dimensional rotated boxes $B_{i,j}$,

$$\Omega_{\text{box}} := \prod_{k \in \mathcal{J}_I} I_k \times \prod_{(i,j) \in \mathcal{J}_{\text{pair}}} B_{i,j},$$

where $\mathcal{J}_I = \{i \in \{1, \dots, d\} : i \text{ is an unpaired dimension}\}$, $\mathcal{J}_B = \{1, \dots, d\} \setminus \mathcal{J}_I$, and $\mathcal{J}_{\text{pair}} = \{\, (i,j) \in \mathcal{J}_B \times \mathcal{J}_B \mid \text{the dimensions } i \text{ and } j \text{ are coupled}\}$.

During the execution of the algorithm, Ω_{box} is represented as a vector-matrix pair $(\mathbf{V}, \mathbf{M})$. The vector $\mathbf{V} \in \mathbb{R}^d$ denotes an arbitrary vertex of the box and acts as origin of the rotated coordinate system described by the edges of the box. The matrix $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_d] \in \mathbb{R}^{d \times d}$ contains the basis of this rotated coordinate system. Each column $\mathbf{m}_1, \dots, \mathbf{m}_d$ of $\mathbf{M}$ is the distance vector from $\mathbf{V}$ to one of its neighbouring vertices. They are ordered such that the edge that is axis-parallel in dimension i occupies the i-th column and the two edges that are associated with a 2D-map $\Omega_{i,j}$ occupy the i-th and j-th column (see the example below).

This representation of rotated boxes turned out the most useful for the following reasons:

- All other vertices of the box can be constructed by a linear combination of $\mathbf{V}$ and the columns of $\mathbf{M}$.
- Representing a d-dimensional rotated box by its vertices would require a list with 2^d entries. For large d, this would be infeasible.[1]
- Any affine transformation of the rotated box can be carried out by applying that transformation to $\mathbf{V}$ and $\mathbf{M}$.

[1]For example, for $d = 100$ spatial dimensions, we would need about $2^{100} \cdot 100 \cdot 64$ Bit $= 10^{23}$ Gigabyte of disk space to store the 2^{100} vertices.

- The intervals I_k and two-dimensional rotated boxes $B_{i,j}$ can be retrieved by setting

$$I_k := \{\, x \in \mathbb{R} \mid x = v_k + t \cdot m_{k,k},\, t \in [0,1] \}$$

and

$$B_{i,j} := \left\{ \boldsymbol{x} \in \mathbb{R}^2 \;\middle|\; \boldsymbol{x} = \begin{bmatrix} v_i \\ v_j \end{bmatrix} + s \cdot \begin{bmatrix} m_{i,i} \\ m_{i,j} \end{bmatrix} + t \cdot \begin{bmatrix} m_{j,i} \\ m_{j,j} \end{bmatrix},\, (s,t) \in [0,1]^2 \right\}.$$

For example, the rotated box given by the vertices $\big(4,2,0\big)$, $\big(4,4,\frac{4}{3}\big)$, $\big(4,2,\frac{13}{3}\big)$, $\big(4,0,3\big)$, $\big(0,2,0\big)$, $\big(0,4,\frac{4}{3}\big)$, $\big(0,2,\frac{13}{3}\big)$, and $\big(0,0,3\big)$, compare Fig. 2, can be expressed by

$$\mathbf{V} = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} -4 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & \frac{4}{3} & -3 \end{bmatrix}.$$

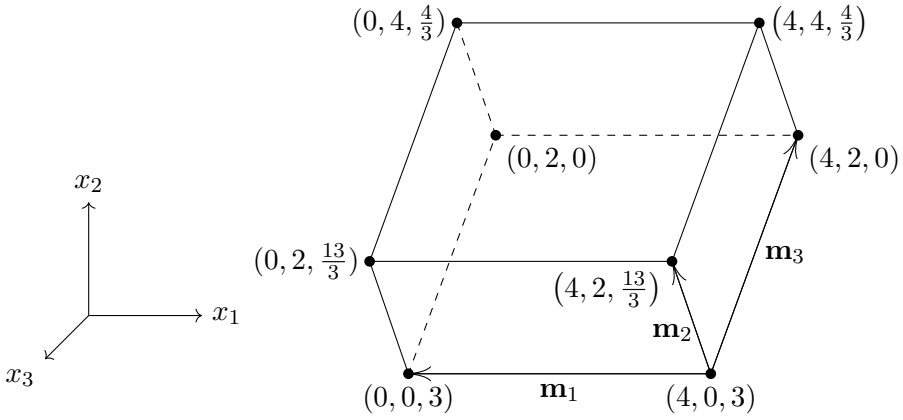

FIGURE 2. A rotated box in three dimensions.

3. Analysis of Covariance

Sometimes it might not be obvious which pairs of design variables should be chosen for technical reasons. One idea to maximize the resulting solution space is to analyse in a first step of the optimization procedure the covariance matrix of good design points. To this end, design points are randomly sampled from the whole design space Ω_{ds} and all the good design points are put into the matrix

$$\mathbf{X}^{\mathrm{good}} = \left[{}_{(1)}\boldsymbol{x}^{\mathrm{good}}, \ldots, {}_{(n^{\mathrm{good}})}\boldsymbol{x}^{\mathrm{good}} \right] \in \mathbb{R}^{n^{\mathrm{good}} \times d}.$$

Then, the corresponding covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ is calculated, with

$$\boldsymbol{\Sigma}_{i,j} = \mathrm{cov}\left(\mathbf{X}_i^{\mathrm{good}}, \mathbf{X}_j^{\mathrm{good}} \right), \quad 1 \le i,j \le d.$$

Here, $\mathbf{X}_i^{\text{good}}$ and $\mathbf{X}_j^{\text{good}}$ denote the i-th and j-th column of the matrix $\mathbf{X}^{\text{good}}$ and the respective covariance $\operatorname{cov}(\mathbf{X}_i^{\text{good}}, \mathbf{X}_j^{\text{good}})$ is defined by

$$\operatorname{cov}(\mathbf{X}_i^{\text{good}}, \mathbf{X}_j^{\text{good}}) = \frac{1}{n^{\text{good}} - 1} \sum_{k=1}^{n^{\text{good}}} \left(\mathbf{X}_{i,k}^{\text{good}} - \mu_i\right) \left(\mathbf{X}_{j,k}^{\text{good}} - \mu_j\right),$$

where

$$\mu_i = \frac{1}{n^{\text{good}}} \sum_{k=1}^{n^{\text{good}}} \mathbf{X}_{i,k}^{\text{good}}, \quad \mu_j = \frac{1}{n^{\text{good}}} \sum_{k=1}^{n^{\text{good}}} \mathbf{X}_{j,k}^{\text{good}}$$

is their mean.

Since the covariance matrix $\mathbf{\Sigma}$ indicates the dimensions that admit a strong coupling, it is useful in determining the choice of design variables for a 2D-map. The 2D-maps $\Omega_{i,j}$ with the strongest coupling of the dimensions i and j can be found iteratively by choosing those pairs (i, j) with $\mathbf{\Sigma}_{i,j} = \max_{(k,\ell)\in\mathcal{I}} |\mathbf{\Sigma}_{k,\ell}|$ and

$$\mathcal{I} = \left\{(k, \ell) \in \{1, \ldots, d\}^2 : k < \ell \text{ and } k, \ell \text{ not part of another 2D-map}\right\}.$$

It should be noted that this way of determining the coupled pairs of design variables is purely mathematical and does not take aspects of design into account.

4. Box-rotation algorithm

The algorithm in this article is based on the same steps as the sampling-based box optimization algorithm and extends it by one step to rotate the candidate box. "Rotate box" is performed after sampling the design points and before trimming. The flowchart in Fig. 3 gives an overview of the box-rotation algorithm, where the "Rotate box" step is highlighted by the red background colour. All steps had to be modified to account for the no longer axis-parallel boxes in the algorithm. A detailed explanation of all the steps is provided in the following subsections.

4.1. **Box initialization.**

The initial box is either given by the problem or can be constructed by using a classical optimization algorithm and building a sufficiently large box around around an optimal design point. Also a genetic algorithm, such like differential evolution for example, cf. [13], can be used. This step is the same as in the box optimization.

4.2. **Exploration phase.**

In the exploration phase, the box moves around through the design space in order to find a large region of good design. This is done by trimming, rotation and growing the box for n^{exp} steps, where n^{exp} is a fixed number given in advance by the user.

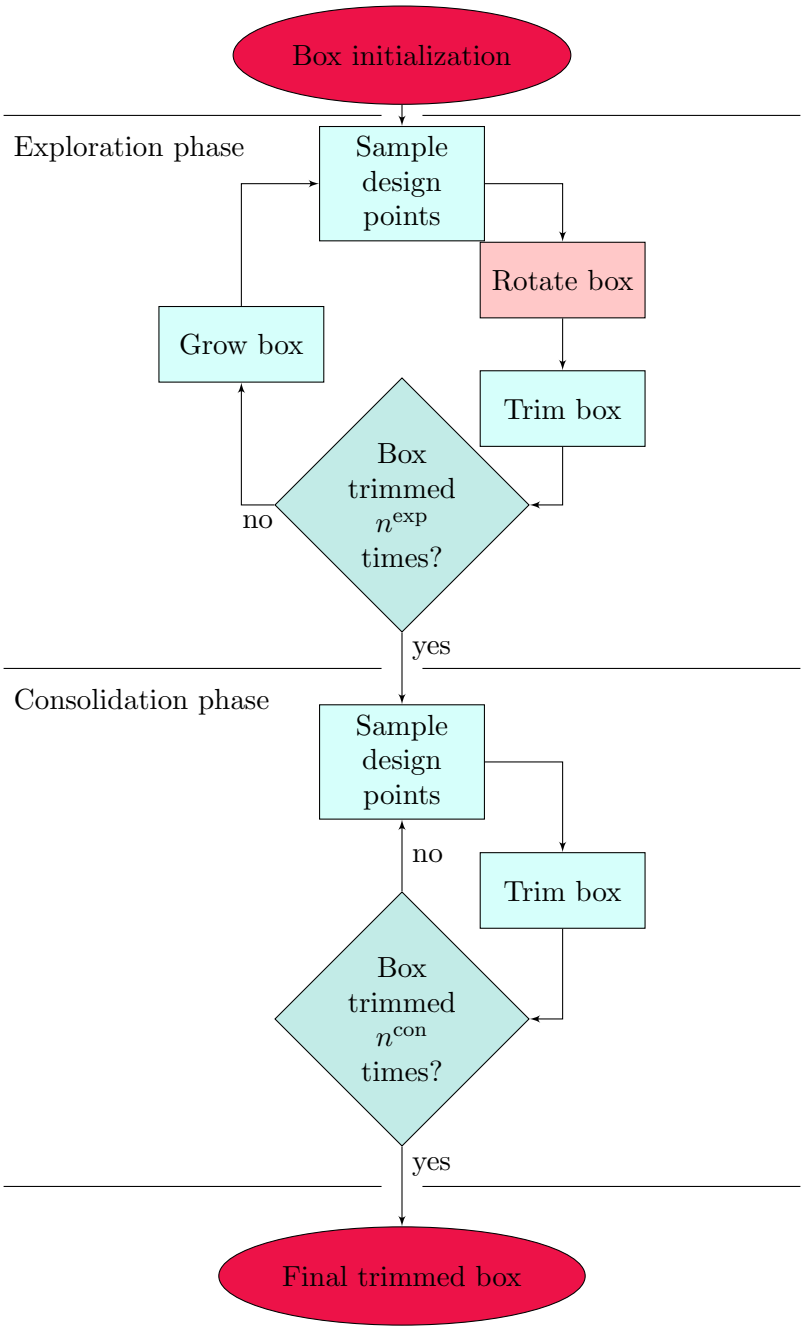

FIGURE 3. The box-rotation algorithm. It coincides with the box optimization if the field "Rotate box", highlighted in light red, is removed.

4.2.1. *Sample design points.* Inside Ω_{box}, N uniformly distributed random design points are sampled.[2] The designs are evaluated by the objective function f and divided into a set $\mathcal{X}^{\text{good}} = \left\{ {}_{(1)}\boldsymbol{x}^{\text{good}}, \ldots, {}_{(n^{\text{good}})}\boldsymbol{x}^{\text{good}} \right\}$ containing all good design points and a set $\mathcal{X}^{\text{bad}} = \left\{ {}_{(1)}\boldsymbol{x}^{\text{bad}}, \ldots, {}_{(n^{\text{bad}})}\boldsymbol{x}^{\text{bad}} \right\}$ containing all bad design points. The elements of these sets are ordered from highest to lowest by their objective value $f(\boldsymbol{x})$. Additionally, all design points are mapped to the d-dimensional unit cube $[0,1]^d$ to normalize all design variables.

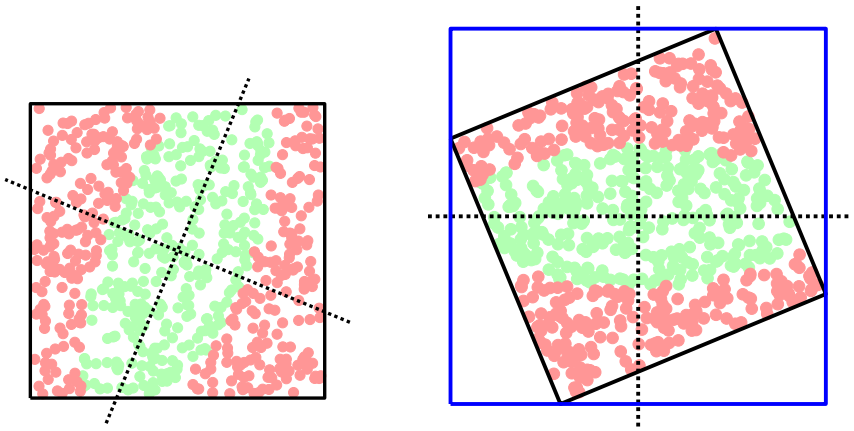

FIGURE 4. The design points and principal components (dotted) before (left) and after (right) rotation. The blue line indicates the axis-parallel bounding box around the rotated box.

4.2.2. *Rotate box.* The optimal box-rotation is determined with the help of the *principal component analysis* (PCA, see [14]) applied to the good design points $\mathcal{X}^{\text{good}}$, projected onto the respective 2D-map $\Omega_{i,j}$. Namely, the coordinate system of $\Omega_{i,j}$ is rotated such that the two dominant principle components form the coordinate axes (see Fig. 4). Ω_{box} and all design points are transformed into the new coordinate systems. They are transformed such that the origin of the rotated coordinate system is the center of gravity of the good design points $\mathbf{X}^{\text{good}}$.

The details of the box-rotation step are shown in the Box-Rotation Algorithm. The first input parameter (see line 1) is $\Omega_{\text{box}} = (\mathbf{V}, \mathbf{M})$. The second and third inputs are two matrices

$$\mathbf{X}^{\text{good}} = \left[{}_{(1)}\boldsymbol{x}^{\text{good}}, \ldots, {}_{(n^{\text{good}})}\boldsymbol{x}^{\text{good}} \right] \in \mathbb{R}^{n^{\text{good}} \times d},$$
$$\mathbf{X}^{\text{bad}} = \left[{}_{(1)}\boldsymbol{x}^{\text{bad}}, \ldots, {}_{(n^{\text{bad}})}\boldsymbol{x}^{\text{bad}} \right] \in \mathbb{R}^{n^{\text{bad}} \times d}.$$

[2]For practical applications, $N = 100$ turned out to be a good choice, compare [13].

These are the sets $\mathcal{X}^{\text{good}}$ and $\mathcal{X}^{\text{bad}}$ written as matrices, where each row contains one design point.

The algorithm begins by iterating over the 2D-maps $\Omega_{i,j}$, see line 3 of the Box-Rotation Algorithm. On each 2D-map, we calculate the principal components of the good design points, which are the eigenvectors of the sample covariance matrix of the good design points on that map (line 4). The procedure `eigenvectors`$(\cdot)$ calculates the *normalized* eigenvectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2]$ of this 2×2 covariance matrix.

In line 5 of the Box-Rotation Algorithm, the mean μ_i and μ_j of the good design points in the dimensions i and j are calculated. Subsequently, in lines 6 and 7, the actual rotation of the design points happens. The means μ_i and μ_j are subtracted from all design points to normalize them with respect to the origin. Then, the design points are multiplied with the matrix $\mathbf{E}$ to rotate them such that the principal components form the new coordinate axes. Finally, in lines 8 and 9, the same rotation to the respective values of the tuple $(\mathbf{V}, \mathbf{M})$ are applied, which determines the solution space Ω_{box} in the dimensions i and j.

Box-Rotation Algorithm. This algorithm rotates the box by rotating the coordinate system.

1: **Input:** Ω_{box}, $\mathbf{X}^{\text{good}}$, $\mathbf{X}^{\text{bad}}$
2: **Output:** Ω_{box}, $\mathbf{X}^{\text{good}}$, $\mathbf{X}^{\text{bad}}$

3: **for all** 2D-maps $\Omega_{i,j}$ **do**
4: $\quad \mathbf{E} \leftarrow \texttt{eigenvectors}\left(\begin{bmatrix} \operatorname{cov}(\mathbf{X}_i^{\text{good}}, \mathbf{X}_i^{\text{good}}) & \operatorname{cov}(\mathbf{X}_i^{\text{good}}, \mathbf{X}_j^{\text{good}}) \\ \operatorname{cov}(\mathbf{X}_j^{\text{good}}, \mathbf{X}_i^{\text{good}}) & \operatorname{cov}(\mathbf{X}_j^{\text{good}}, \mathbf{X}_j^{\text{good}}) \end{bmatrix}\right)$
5: $\quad [\mu_i, \mu_j] \leftarrow \frac{1}{n^{\text{good}}} \sum_{k=1}^{n^{\text{good}}} [x_{k,i}, x_{k,j}]$
6: $\quad \left[\mathbf{X}_i^{\text{good}}, \mathbf{X}_j^{\text{good}}\right] \leftarrow \left[\mathbf{X}_i^{\text{good}} - \mu_i, \mathbf{X}_j^{\text{good}} - \mu_j\right] \cdot \mathbf{E}^\top$
7: $\quad \left[\mathbf{X}_i^{\text{bad}}, \mathbf{X}_j^{\text{bad}}\right] \leftarrow \left[\mathbf{X}_i^{\text{bad}} - \mu_i, \mathbf{X}_j^{\text{bad}} - \mu_j\right] \cdot \mathbf{E}^\top$
8: $\quad \left[v_i, v_j\right] \leftarrow \left[v_i - \mu_i, v_j - \mu_j\right] \cdot \mathbf{E}^\top$
9: $\quad \begin{bmatrix} m_{i,i} & m_{i,j} \\ m_{j,i} & m_{j,j} \end{bmatrix} \leftarrow \begin{bmatrix} m_{i,i} & m_{i,j} \\ m_{j,i} & m_{j,j} \end{bmatrix} \cdot \mathbf{E}^\top$
10: **end for**

4.2.3. *Trim box.* In this part of the algorithm, the trimming from the box optimization is applied to the design points in the rotated coordinate system. However, a few adjustments have to be made before the box can actually be trimmed.

With respect to the new coordinate system, determined in Subsection 4.2.2, the solution space Ω_{box} is still a product of one-dimensional intervals I_k and two-dimensional rotated boxes $B_{i,j}$. Nonetheless, the original trimming from

[13] is only applicable to axis-parallel boxes. Therefore, Ω_{box} is modified by constructing a bounding box around each rotated box $B_{i,j}$, that is

$$B_{i,j} \subset \left[\widetilde{a}_i, \widetilde{b}_i\right] \times \left[\widetilde{a}_j, \widetilde{b}_j\right],$$

where

$$\left.\begin{aligned} \widetilde{a}_\ell &= \min\{x_\ell : (x_i, x_j) \in B_{i,j}\} \\ \widetilde{b}_\ell &= \max\{x_\ell : (x_i, x_j) \in B_{i,j}\} \end{aligned}\right\} \quad \ell = i, j.$$

This yields a new box that is axis-parallel in the rotated coordinate system (compare Fig. 4):

$$\widetilde{\Omega}_{\text{box}} = \prod_{i=1}^{d} \left[\widetilde{a}_i, \widetilde{b}_i\right].$$

Now, the original trimming algorithm can be applied to this box, resulting in a box $\Omega^\star_{\text{box}}$ that is axis-parallel in the rotated coordinate system. With respect to the original coordinate system, the box has seemingly been rotated such that it lies in those directions of the good design points that maximize their variance. The pseudo code is given in the Box Trimming Algorithm. Note that the notation from [11] has been modified such that it matches this article.

Box Trimming Algorithm. This algorithm trims the box such that the smallest number of good design points is removed.

Input: Ω_{box}, $\mathcal{X}^{\text{good}}, \mathcal{X}^{\text{bad}}$
Output: $\Omega^\star_{\text{box}}$

for all $\boldsymbol{x}^{\text{good}} \in \mathcal{X}^{\text{good}}$ **do**
  $\Omega^\star_{\boldsymbol{x}^{\text{good}}} = \prod_{i=1}^{d} [a_i^\star, b_i^\star] \leftarrow \Omega_{\text{box}}$
  for all $\boldsymbol{x}^{\text{bad}} \in \mathcal{X}^{\text{bad}}$ **do**
    $[\boldsymbol{n}^{\text{good}}, \boldsymbol{n}^{\text{bad}}] = \texttt{countpoints}(\boldsymbol{x}^{\text{good}}, \boldsymbol{x}^{\text{bad}}, \Omega_{\text{box}}, \mathcal{X}^{\text{good}}, \mathcal{X}^{\text{bad}})$
    $I_{\text{good}} \leftarrow \left\{ i \in \{1, \ldots, d\} \;\middle|\; n_i^{\text{good}} = \min_{j \in \{1,\ldots,d\}} n_j^{\text{good}} \right\}$
    $I_{\text{bad}} \leftarrow \left\{ i \in I_{\text{good}} \;\middle|\; n_i^{\text{bad}} = \max_{j \in I_{\text{good}}} n_j^{\text{bad}} \right\}$
    $i^\star \in_{\text{rand}} I_{\text{bad}}$
    if $x_{i^\star}^{\text{bad}} < x_{i^\star}^{\text{good}}$ **then** $a_{i^\star}^\star \leftarrow x_{i^\star}^{\text{bad}}$ **else** $b_{i^\star}^\star \leftarrow x_{i^\star}^{\text{bad}}$ **end if**
  end for
  $\mathcal{X}_{\cap}^{\text{good}} \leftarrow \mathcal{X}^{\text{good}} \cap \prod_{i=1}^{d} [a_i^\star, b_i^\star]$
  for all $a_i^\star \neq a_i$ **do** $a_i^\star \leftarrow \min_{\boldsymbol{x}^{\text{good}} \in \mathcal{X}_{\cap}^{\text{good}}} x_i^{\text{good}}$ **end for**
  for all $b_i^\star \neq b_i$ **do** $b_i^\star \leftarrow \max_{\boldsymbol{x}^{\text{good}} \in \mathcal{X}_{\cap}^{\text{good}}} x_i^{\text{good}}$ **end for**
end for
$\Omega^\star_{\text{box}} \leftarrow \arg\max_{\Omega^\star_{\boldsymbol{x}^{\text{good}}}} \mu\left(\Omega^\star_{\boldsymbol{x}^{\text{good}}}\right)$

The Box Trimming Algorithm requires an axis-parallel solution space $\Omega_{\text{box}} = \prod_{i=1}^{d} [a_i, b_i]$ as well as sets of good and bad design points $\mathcal{X}^{\text{good}}$ and $\mathcal{X}^{\text{bad}}$ as

inputs (line 1). The output (line 2) is a trimmed box $\Omega^\star_{\text{box}} = \prod_{i=1}^d [a_i^\star, b_i^\star]$. Ω_{box} is trimmed by moving its boundaries onto the bad design points until there are only good design points left. Because there is no unique way to do this, multiple boxes $\Omega^\star_{\text{box}}$ are calculated, such that for each good design point $\boldsymbol{x}^{\text{good}}$ there exists at least one trimmed box $\Omega^\star_{\boldsymbol{x}^{\text{good}}}$ that contains it, cf. [11]. From those boxes, the one with the largest volume is chosen as the final box $\Omega^\star_{\text{box}}$.

The Box Trimming Algorithm iterates therefore over all good design points $\boldsymbol{x}^{\text{good}}$ (line 3), initializes a new box $\Omega^\star_{\boldsymbol{x}^{\text{good}}}$ in line 4, and then begins a loop over the bad design points $\boldsymbol{x}^{\text{bad}}$ in line 5. For each bad design point, it counts how many design points would get removed if the box is trimmed to $\boldsymbol{x}^{\text{bad}}$ in line 6, using the procedure `countdesign points`(), introduced in the Count Points Algorithm and described below. Then, it finds the dimensions where the fewest good design points are removed (line 7), chooses from those the dimensions where the most design points are removed (line 8), and finally, if it has not found a unique dimension yet, chooses one of those dimensions at random (line 9). In line 10, the box is trimmed in the chosen dimension to the current bad design point such that the current good design point does not get removed.

After having iterated over all bad design points, the remaining good design points are gathered (line 12) and the boundaries are trimmed further to the nearest good design points in dimensions where the boundaries actually had to be trimmed (lines 13 and 14). Finally, after iterating over all good design points, the box $\Omega^\star_{\boldsymbol{x}^{\text{good}}}$ with the highest volume μ is chosen as the output (line 16).

The procedure `countpoints()`, as implemented in Count Points Algorithm, counts the removed design points if the box is trimmed to x_i^{bad} in dimension i. It takes a good design point $\boldsymbol{x}^{\text{good}}$, a bad design point $\boldsymbol{x}^{\text{bad}}$, a solution space Ω_{box}, and two sets of good and bad design points as inputs. The outputs are two vectors $\boldsymbol{n}^{\text{good}}$ and $\boldsymbol{n}^{\text{bad}}$ that count how many good and bad design points get removed if the boundary of Ω_{box} in dimension i is moved onto x_i^{bad}, but the design point $\boldsymbol{x}^{\text{good}}$ is left inside Ω_{box}. To this end, the algorithm iterates over all dimensions (see line 3), and decides whether $x_i^{\text{bad}} < x_i^{\text{good}}$ or $x_i^{\text{bad}} \geq x_i^{\text{good}}$ (lines 4 and 7). Then, it counts the number of design points between the boundary and x_i^{bad} (lines 5–9).

4.2.4. *Grow box.* The last part of one exploration step k is growing the box. Each interval $[a_i, b_i]$ is stretched by a growth rate $g^{(k)}$ such that

$$\Omega_{\text{box}} = \prod_{i=1}^{d} \left[\overline{a}_i, \overline{b}_i\right]$$

with

$$\overline{a}_i := a_i - g^{(k)} \cdot (b_i - a_i), \quad \overline{b}_i := b_i + g^{(k)} \cdot (b_i - a_i).$$

Count Points Algorithm. This algorithm counts the design points which are removed by the trimming step.

Input: $\boldsymbol{x}^{\text{good}}, \boldsymbol{x}^{\text{bad}}, \Omega_{\text{box}}, \mathcal{X}^{\text{good}}, \mathcal{X}^{\text{bad}}$
Output: $\boldsymbol{n}^{\text{good}}, \boldsymbol{n}^{\text{bad}}$

for $i = 1, \ldots, d$ **do**
  if $x_i^{\text{bad}} < x_i^{\text{good}}$ **then**
    $n_i^{\text{good}} \leftarrow \#\left\{ \boldsymbol{x} \in \mathcal{X}^{\text{good}} \;\middle|\; a_i \le x_i \le x_i^{\text{bad}} \right\}$
    $n_i^{\text{bad}} \leftarrow \#\left\{ \boldsymbol{x} \in \mathcal{X}^{\text{bad}} \;\middle|\; a_i \le x_i \le x_i^{\text{bad}} \right\}$
  else
    $n_i^{\text{good}} \leftarrow \#\left\{ \boldsymbol{x} \in \mathcal{X}^{\text{good}} \;\middle|\; x_i^{\text{bad}} \le x_i \le b_i \right\}$
    $n_i^{\text{bad}} \leftarrow \#\left\{ \boldsymbol{x} \in \mathcal{X}^{\text{bad}} \;\middle|\; x_i^{\text{bad}} \le x_i \le b_i \right\}$
  end if
end for

The growth rate may either be a constant $g^{(0)}$ set at the beginning of the algorithm, such that

$$g^{(k)} := g^{(k-1)},$$

or, before growing the box, it may be updated according to the formula

$$g^{(k)} := \frac{a_k^{\text{good}}}{a^{\text{target}}} \cdot g^{(k-1)},$$

where $a_k^{\text{good}} = n_k^{\text{good}}/N$ is the fraction of good design points before trimming the box in exploration step k and a^{target} is the desired fraction of good design points.

Note that the box may grow into the exterior of Ω_{ds}, including design variables that are out of scope. In the box optimization, this has been solved by simply retracting the axis-parallel box onto the boundary of Ω_{ds}. It did not lose any good design space in the process. However, if a rotated box has to be retracted into Ω_{ds}, a large amount of the good design space that has already been found could be lost, simply by trying to fit the box in Ω_{ds}. Thus, without further change of the notation, the following two modifications are introduced:

(1) In each step "Sample design points", design points are sampled within $\Omega_{\text{box}} \cap \Omega_{\text{ds}}$ instead of only Ω_{box}.
(2) In each step "Trim box", when the volume $\mu\left(\Omega^\star_{\boldsymbol{x}^{\text{good}}}\right)$ is calculated, the volume $\mu\left(\Omega^\star_{\boldsymbol{x}^{\text{good}}} \cap \Omega_{\text{ds,rot}}\right)$ is calculated instead, where $\Omega_{\text{ds,rot}}$ is the transformation of $[0,1]^d$ (which is the normalized design space) into the rotated coordinate system prescribed by the 2D-maps,

$$\Omega_{\text{ds,rot}} = \prod_{k \in \mathcal{I}_I} [0,1] \times \prod_{(i,j) \in \mathcal{J}_{\text{pair}}} \rho_{i,j}\left([0,1]^2\right),$$

and $\rho_{i,j}$ is the coordinate transform of the map $\Omega_{i,j}$ in the box-rotation step.

The box is not grown in the final step n^{exp} of the exploration phase. This growth step is skipped and the algorithm switches to the *consolidation phase*.

4.3. **Consolidation phase.** The box is no longer rotated or grown in this phase, since it is expected to be in a position with a large amount of good design space. However, the box might still contain some bad design space, and the goal of this phase is to remove as much bad design space as possible. Thus, one step of the consolidation phase consists of sampling design points and then trimming the box. The consolidation phase is terminated after either a fixed number of n^{con} steps or when no bad design points have been sampled three times in series. Terminating after three of those steps is done to save time, the reasoning being that more consolidation steps change little in the quality of the box. The final solution space of the consolidation phase is taken as the result of the box-rotation algorithm.

5. Numerical experiments

5.1. **Problem 1: Diagonal solution space.** It is clear that the principle component analysis converges to the correct angle if the sample size is increased. In order to verify the convergence of the entire box-rotation algorithm, we test whether the angles of the rotated boxes converge to the correct angle. To this end, the design space $\Omega_{\mathrm{ds}} = [0,1]^2$ is considered and the objective function is defined such that the good solution space generates a diagonal corridor of width $\frac{1}{\sqrt{2}}$ in the design space. The angle between the corridor and the x-axis is $45°$, see Fig. 5 for an illustration. The growth of the box is dynamic, with $a^{\mathrm{target}} = 0.8$ and an initial growth rate of 0.1.

The algorithm is repeated 100 times each for the different sample sizes $N = 50, 60, 70, \ldots, 500$. For each final box, the angle between the box and the x-axis is calculated. The mean of the angles is close to $45°$ for all sample sizes, and the standard deviation of $2.5°$ does not change significantly for sample sizes larger than about 200. The observed results form a funnel, compare Fig. 6. This can be interpreted as a sign that, with growing sample sizes, the angle of the final rotated box converges to $45°$.

Next, the box-rotation algorithm is tested for different corridor widths, i.e. $0.05, 0.1, \ldots, 0.75$, with the number of design points fixed to 100. The algorithm is executed 100 times for each width, skipping the consolidation phase in every execution.

As can be seen in Fig. 7, the angle of the box varies the more the wider the corridors become. This is due to the fact that, for a large corridor size, the box taps into the region outside the design space, where no design points are sampled. Thus, the final position of the box varies a lot more. This

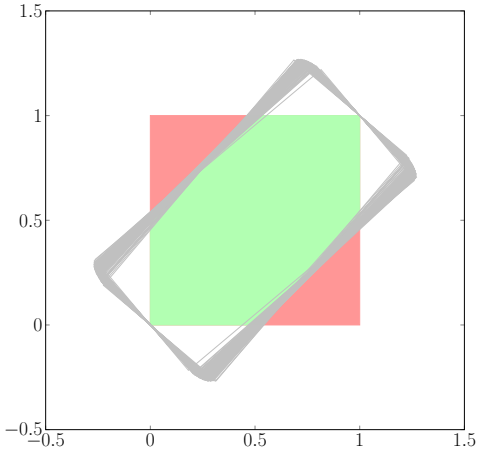

FIGURE 5. 100 solution boxes for the sample size $N = 500$ for problem 1: diagonal solution space.

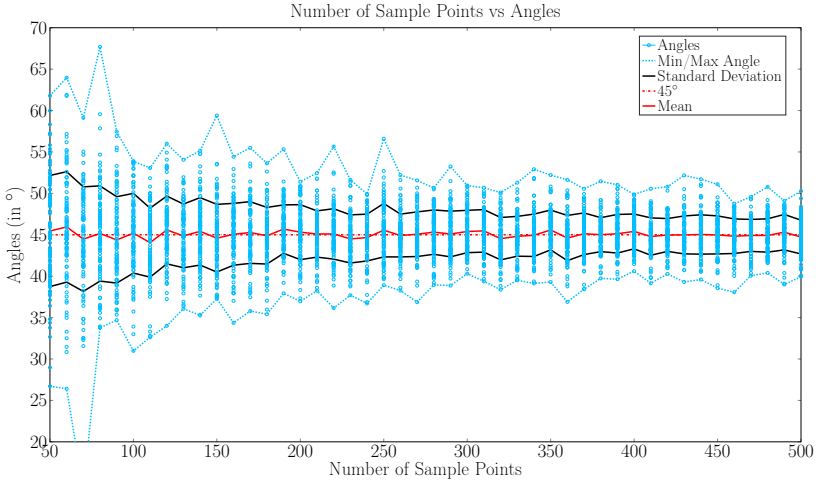

FIGURE 6. Funnels sample sizes from 50 to 500.

observation is confirmed by Fig. 8, where the solution boxes are found for the corridor width 0.1 (left plot) and the corridor width 0.75. The non-defined space contained in the solution box is much larger in the right plot than in the left plot. However, the mean stays very close to 45° throughout the whole test, and the standard deviation is only about 5° for a corridor of width 0.75, compare Fig. 7. This suggests that the resulting boxes again converge to 45° and the algorithm works as intended.

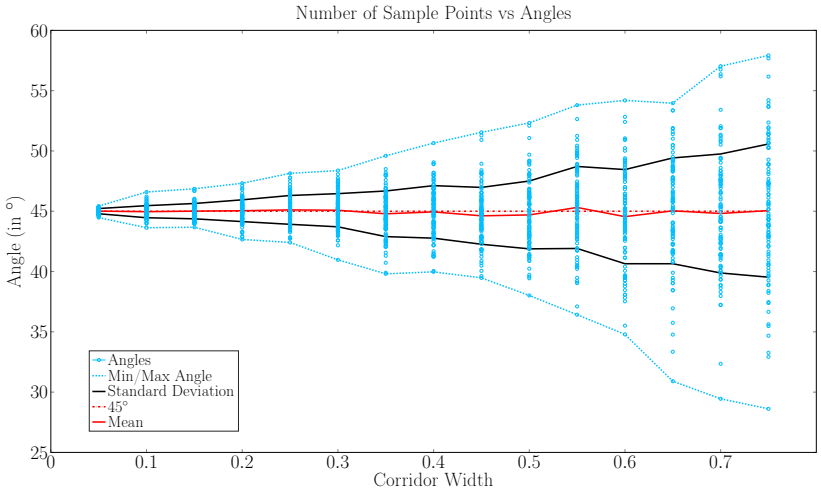

FIGURE 7. The test results for a varying corridor widths.

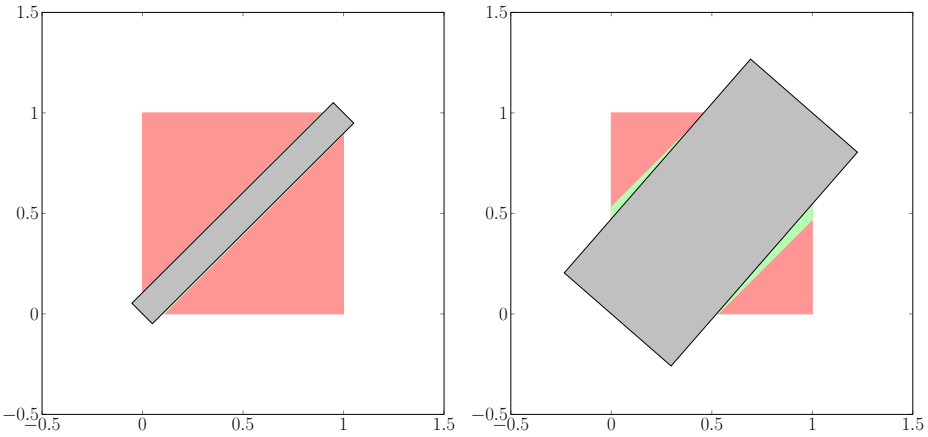

FIGURE 8. The final box for a corridor of width 0.15 (left) and for a corridor of width 0.75 (right).

5.2. **Problem 2: 6D linear problem from vehicle dynamics.** The solutions of the box optimization and the box-rotation algorithm are compared. To that end, the algorithms are applied to a chassis design problem. A similar problem with more dimensions has been analyzed in [9]. The problem at hand is considered relevant because it is derived from a real-world application. Some pairs of design variables have a strong physical interaction with each other, which yields a natural choice for the 2D-maps. Based on this, one can also expect that the good solution space is skewed in the coupled dimensions, which in turn should favour the box-rotation algorithm over the box optimization. In this problem, the design space is

$\Omega_{\text{ds}} = [0.5, 3]^2 \times [0.6, 1.4]^2 \times [0.5, 2]^2$. The design variables are the scaling factors of certain vehicle characteristics. Their descriptions and the intervals assigned to them can be found in Table 1.

Design Variable	Interval	Scaling Factor
x_1	[0.5,3]	force-velocity characteristics of the dampers of the front axle
x_2	[0.5,3]	force-velocity characteristics of the dampers of the rear axle
x_3	[0.6,1.4]	force-displacement of the bearing spring of the front axle
x_4	[0.6,1.4]	force-displacement of the bearing spring of the rear axle
x_5	[0.5,2]	stiffness of the anti-roll bar of the front axle
x_6	[0.5,2]	stiffness of the anti-roll bar of the rear axle

TABLE 1. Design variables for Problem 2.

The design variables for all other components (shock absorbers, tires, etc.) are assumed to already have been fixed earlier in the development process. They are not influenced by the choice of the design variables $x_1, \ldots, x_6$.

The linear objective function $f : \Omega_{\text{ds}} \to \mathbb{R}$ is obtained by the following procedure: Let

$$\boldsymbol{g}_c = \left[0.421, -0.054, 0.414, 0.724, 0.243, 0.027, 0.371\right]^\top$$

and

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0.036 & -0.203 & 0.258 & -0.102 \\ 0 & 0 & 0.111 & -0.313 & 0.540 & -0.165 \\ 0 & 0 & -0.115 & 0.273 & -0.495 & 0.143 \\ -0.152 & -0.025 & -0.078 & 0.121 & -0.329 & 0.057 \\ -0.020 & 0.216 & -0.148 & 0.339 & -0.586 & 0.155 \\ 0 & 0 & 0.088 & 0.144 & 0.391 & 0.072 \\ 0 & 0 & -0.088 & -0.144 & -0.391 & -0.072 \end{bmatrix}.$$

Given $\boldsymbol{x} \in \Omega_{\text{ds}}$, normalize $\boldsymbol{x}$ with respect to the cube $[-1, 1]^6$ in accordance with

$$x_i^{norm} = 2\frac{x_i - x_i^{low}}{x_i^{up} - x_i^{low}} - 1, \quad i = 1, \ldots, 6,$$

where $\boldsymbol{x}^{low} = [0.5, 0.5, 0.6, 0.6, 0.5, 0.5]^\top$ and $\boldsymbol{x}^{up} = [3, 3, 1.4, 1.4, 2, 2]^\top$. Then, apply the matrix $\mathbf{G}$ such that

$$\boldsymbol{y} := \mathbf{G}\boldsymbol{x}^{norm}.$$

	Absolute Volume	Normalized Volume
Box Optimization	0.0035	$3.9 \cdot 10^{-4}$
Box-Rotation Algorithm	0.014	$1.6 \cdot 10^{-3}$

TABLE 2. Results from a single experiment.

Finally, the objective function is

$$f(\boldsymbol{x}) = \begin{cases} 0, & \text{if } y_i \leq g_c^{(i)} \text{ for all } i = 1, \dots, 6, \\ 1, & \text{otherwise,} \end{cases}$$

with the critical threshold $c = 0.5$. Additionally, the growth of the box is again dynamic, with $a^{\text{target}} = 0.8$ and an initial growth rate of 0.1.

Since the problem is linear, the largest possible solution space is determined by means of the linear solution space algorithm proposed in [9]. Its volume is 0.025 and the *normalized* volume is $2.8 \cdot 10^{-3}$, the latter of which is obtained by rescaling Ω_{ds} onto the six-dimensional unit cube $[0,1]^6$. The same initial box and growth parameters are used for both the box optimization and the box-rotation algorithm. For the box-rotation algorithm, those design variables are coupled that describe the same component of the vehicle on 2D-maps. This results in the 2D-maps $\Omega_{1,2} = [0.5, 3]^2$, $\Omega_{3,4} = [0.6, 1.4]^2$ and $\Omega_{5,6} = [0.5, 2]^2$.

The results are found in Fig. 9. The solution space found by the linear solution space algorithm on each 2D-map is drawn and green. The space drawn in red is discarded by the linear solution space algorithm, but may still contain good designs. The solution boxes of the two other algorithms, drawn in grey, are laid over the solution space given by the linear solution space algorithm. As can be clearly seen, the rotated box is larger than the axis-parallel box on every 2D-map. Additionally, the rotated box adjusts to the solution space found by the linear algorithm. As can be seen from Table 2, the rotated box optimization computes a solution box that is about three times larger compared to the box optimization.

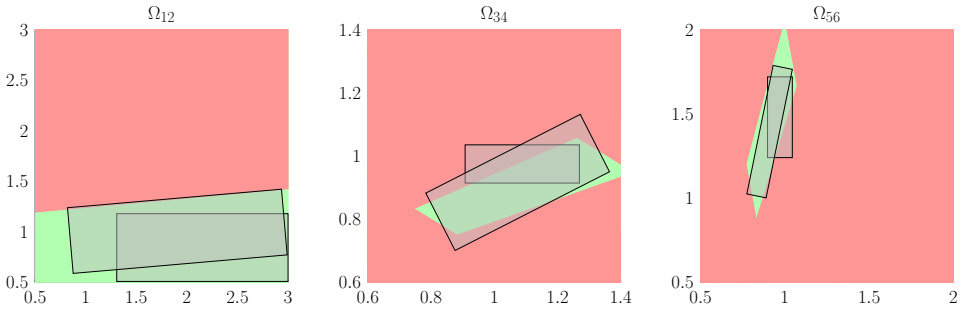

FIGURE 9. The final solution boxes for the 6D linear problem.

This experiment is repeated 100 times for the box optimization and for the box-rotation algorithm, respectively. Then, the mean of the normalized volumes of the computed solution spaces is calculated for both algorithms. The result for the box algorithm is $2.7 \cdot 10^{-3}$ and the result for the box-rotation algorithm is $1.6 \cdot 10^{-3}$. Therefore, the conclusion is that the box-rotation algorithm is clearly superior to the box optimization when trying to maximize the size of the box.

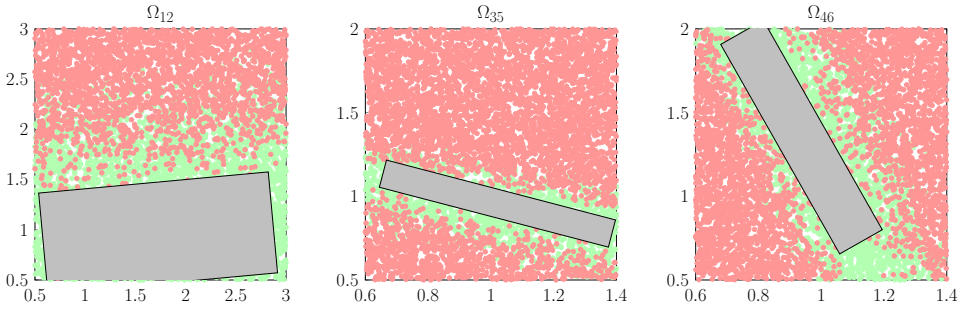

FIGURE 10. The final solution box for the 6D linear problem with an a-priori analysis of covariance, volume $= 6.9 \cdot 10^{-3}$.

Finally, the covariance of the good design points from a sample of 100 design points are analyzed before running the experiment (see Section 3). The 2D-maps proposed by this analysis are $\Omega_{1,2} = [0.5, 3]^2$ and $\Omega_{3,5} = \Omega_{4,6} = [0.6, 1.4] \times [0.5, 2]$. Using these 2D-maps, the experiment is again repeated 100 times for the box-rotation algorithm. The mean volume of all boxes is $3.7 \cdot 10^{-3}$, which is significantly more volume than what is achieved with the given 2D-maps $\Omega_{1,2}$, $\Omega_{3,4}$ and $\Omega_{5,6}$, regardless of the algorithm. It is worthy to note that this increase in volume is achieved with a low computational cost, since the objective function f has to be evaluated only a hundred times more, which is equivalent to making one more step in the exploration or the consolidation phase. Additionally, it is interesting to see that the analysis of covariance couples the design variables by where they act in the vehicle (x_3 and x_5 both act on the front axle, where x_4 and x_6 both act on the rear axle) instead of coupling the design variables that belong to the same component. A visualization of the new 2D-maps can be found in Fig. 10.

Note that the visualization for this figure (and the following figures) is different from that in Fig. 9. For each 2D-map $\Omega_{i,j}$, 10 000 design points $\boldsymbol{x} = (x_k)$ are sampled such that $x_k \in \Omega_{\text{box}}$ for $k \neq i, j$ and $x_k \in \Omega_{i,j}$ for $k = i, j$. Thus, each design point $\boldsymbol{x}$ lies within Ω_{box}, except for its entries x_i and x_j, which may lie anywhere in the 2D-map $\Omega_{i,j}$. That way, the design space surrounding Ω_{box} from "inside" Ω_{box} is seen, especially whether the result is acceptable or not. It can be concluded from Fig. 10 that the algorithm finds a region of good designs and fills it out very well since Ω_{box} lies close to bad design space.

5.3. **Problem 3: 8D nonlinear problem from acoustics.** Consider a simple acoustics engineering problem where nine different transfer paths of noise are to be designed such that the total noise level does not exceed a critical threshold value. Each source emits noise with a complex-valued transfer function

$$A_k(\omega)e^{i\phi_k(\omega)}$$

adding up to the total noise level

$$f(\boldsymbol{x}) = \left|x_1 e^{ix_2} + x_3 e^{ix_4} + \cdots + x_{17} e^{ix_{18}}\right|.$$

Note that the dependency on the frequency ω is dropped, since the analysis is carried out for each relevant frequency. Additionally, all the amplitudes are set to a constant value, i.e. $x_1 = x_3 = \cdots = x_{17} = 1$, and the frequency x_2 is fixed to $x_2 = 0$. Altogether, the objective function in this problem is

$$f(\boldsymbol{x}) = \left|1 + \sum_{\ell=1}^{8} e^{ix_\ell}\right|.$$

with $\Omega_{\mathrm{ds}} = [0, 2\pi]^8$ and $c = 1.5$. With these settings, the good solution space has a symmetrical, but otherwise very irregular shape (compare Fig. 11).

This analysis is relevant for vehicle design for acoustic performance. Here, sound travels along several separate transfer paths in the vehicle body, each associated with different components. The total sound pressure perceived by a passenger is the sum of these separate sound waves that may add up or cancel each other depending on their relative phase angle. In the analysis carried out here, phase angles are the design variables that are pair-wise coupled for larger solution spaces. Permissible regions of phase angles serve as design goal for component designers.

Again, the box optimization and the box-rotation algorithm are applied 100 times to the problem. Again, the growth of the box is dynamic, with $a^{\mathrm{target}} = 0.8$ and an initial growth rate of 0.1. Two exemplary boxes are found in Fig. 11. Especially, it can be seen that the problem under consideration leads to highly non-linear interfaces between the areas of good and bad design points.

There is also evidence of multiple regions of good design space (see the 2D-map $\Omega_{7,8}$ in Fig. 11). Dependent on the initial box position and the growth rate, the box optimization as well as the rotated box optimization may move into either of these two regions. If the growth rate is small, the algorithms will gravitate towards the region closest to the initial position of the box. If the growth rate is sufficiently large, the algorithm will move randomly into the region where more good design points are sampled. Because of this, the solutions of both algorithms are only locally optimal.

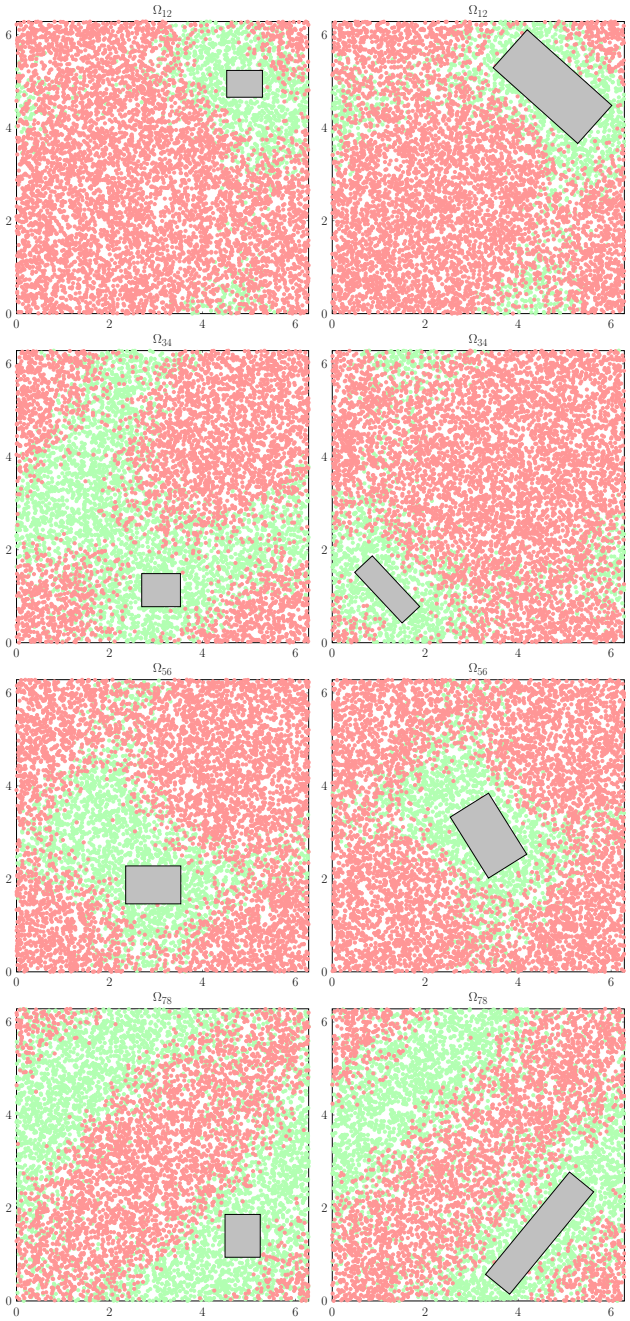

FIGURE 11. Axis-parallel box (left) and rotated box (right) for the 8D problem; the normalized volume is $7.3 \cdot 10^{-8}$ for the axis-parallel box and $2.5 \cdot 10^{-6}$ for the rotated box.

For the box-rotation algorithm, the design variables on the 2D-maps $\Omega_{1,2} = \Omega_{3,4} = \Omega_{5,6} = \Omega_{7,8} = [0, 2\pi]$ are coupled. The mean volume for the normal

box optimization is $8.4 \cdot 10^{-8}$, while the mean volume for the rotated box optimization is $2.9 \cdot 10^{-6}$. This is a huge increase of 35 times more volume.

6. Conclusion

In the present article, the box optimization from [23] has been extended such that it is able to include rotated rectangles. This algorithmic modification can be realized by adding the step "Rotate box" to the box optimization. The concept of 2D-maps introduces a weak coupling to the design variables, but it greatly improves the algorithm in the sense that the final solution box has a much larger volume. Numerical results demonstrate the effectiveness of this approach.

Conflict of interest. On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results. The results presented in this article can be replicated by implementing the data structures and algorithms presented in this article. The objective function of each problem can be used as described in section 5 and only requires the inputs specified in the description of that problem.

References

[1] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer International Publishing AG, Cham, 2017.

[2] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MOS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics and Mathematical Programming Society, Philadelphia, 2001.

[3] H.-G. Beyer and B. Sendhoff. Robust optimization — A comprehensive survey. *Comput. Meth. Appl. Mech. Eng.*, 196(33–34):3190–3218, 2007.

[4] F. Campolongo, M. Ratto, A. Saltelli, and S. Tarantola. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. John Wiley & Sons, Chichester, 2004.

[5] K.K. Choi, B.D. Youn, and R.-J. Yang. Moving least square method for reliability-based design optimization. *Fourth World Congress of Structural and Multidisciplinary Optimization*, Dalian, China, 2001.

[6] P.K. Das, D. Faulkner, and Y. Pu. A strategy for reliability-based optimization. *Engineering Structures*, 19(3):276–282, 1997.

[7] F. Duddeck and E. Wehrle. Recent advances on surrogate modeling for robustness assessment of structures with respect to crashworthiness requirement. *10th European LS-DYNA Conference*, Würzburg, Germany, 2015.

[8] M. Eichstetter, S. Müller, and M. Zimmermann. Product family design with solution spaces. *J. Mech. Design*, 137(12):121401, 2015.

[9] S. Erschen, F. Duddeck, M. Gerdts, and M. Zimmermann. On the optimal decomposition of high-dimensional solution spaces of complex systems. *ASME J. Risk Uncertainty Part B*, 4(2):021008, 2017.

[10] W. Graf, M. Götz, and M. Kaliske. Computing permissible design spaces under consideration of functional responses. *Adv. Eng. Softw.*, 117:95–106, 2018.

[11] L. Graff. *A stochastic algorithm for the identification of solution spaces in high-dimensional design spaces.* PhD thesis, Faculty of Science, University of Basel, 2013.
[12] L. Graff, J. Fender, H. Harbrecht, and M. Zimmermann. Identifying key parameters for design improvement in high-dimensional systems with uncertainty. *J. Mech. Design*, 136(4):041007, 2014.
[13] L. Graff, H. Harbrecht, and M. Zimmermann. On the computation of solution spaces in high dimensions. *Struct. Multidiscip. Optim.*, 54(4):811–829, 2016.
[14] I.T. Jolliffe. *Principal Component Analysis.* Springer, New York, 1986.
[15] R.J. Malak Jr., J.M. Aughenbaugh, and Ch.J.J. Paredis. Multi-attribute utility analysis in set-based conceptual design *Computer-Aided Design*, 41:214227, 2009.
[16] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer J.*, 7(4): 308–313, 1965.
[17] D.J. Singer, N. Doerry, and M.E. Buckley. What is set-based design? *Naval Eng. J.*, 121(4):31–43, 2009.
[18] D. Sobek, A.C. Ward, and J. Liker. Toyota's principles of set-based concurrent engineering. *Sloan Management Review*, 40(2):67–83, 1999.
[19] J. Nocedal and S.J. Wright. Numerical Optimization. Springer, New York, 2006.
[20] R. Storn and K. Price. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.*, 11:341–359, 1997.
[21] J. Tu, K.K. Choi, and Y.H. Park A new study on reliability-based design optimization. *J. Mech. Design*, 121(4):557–564, 1999.
[22] B.D. Youn, K.K. Choi, R.-J. Yang, and L. Gu. Reliability-based design optimization for crashworthiness of vehicle side impact. *Struct. Multidisc. Optim.*, 26:272-283, 2004.
[23] M. Zimmermann and J.E. von Hoessle. Computing solution spaces for robust design. *Int. J. Numer. Meth. Eng.*, 94(3):290–307, 2013.
[24] M. Zimmermann, S. Königs, C. Niemeyer, J. Fender, C. Zeherbauer, R. Vitale, and M. Wahle. On the design of large systems subject to uncertainty. *J. Eng. Design*, 28(4),233–254, 2017.

Helmut Harbrecht, Departement Mathematik und Informatik, Universität Basel, Spiegelgasse 1, 4051 Basel, Schweiz. Fon: +41 (0)61 207 39 92, Fax: +41 (0)61 207 26 95

E-mail address: helmut.harbrecht@unibas.ch

Dennis Tröndle, Departement Mathematik und Informatik, Universität Basel, Spiegelgasse 1, 4051 Basel, Schweiz.

E-mail address: dennis.troendle@unibas.ch

Markus Zimmermann, Lehrstuhl für Produktentwicklung, Technische Universität München, Boltzmannstr. 15, 85748 Garching, Deutschland.

E-mail address: zimmermann@tum.de