ON THE COMPUTATION OF SOLUTION SPACES IN HIGH DIMENSIONS

L. GRAFF, H. HARBRECHT, AND M. ZIMMERMANN

Abstract. A stochastic algorithm that computes box-shaped solution spaces for nonlinear, high-dimensional and noisy problems with uncertain input parameters has been proposed in [35]. This paper studies in detail the quality of the results and the efficiency of the algorithm. Appropriate benchmark problems are specified and compared with exact solutions that were derived analytically. The speed of convergence decreases as the number of dimensions increases. Relevant mechanisms are identified that explain how the number of dimensions affects the performance. The optimal number of sample points per iteration is determined in dependence of the preference for fast convergence or a large volume.

1. Introduction

In many engineering problems, uncertainty is naturally present, especially in the early development phase. Uncertainty arises because some parameters cannot yet exactly be specified or they may be changed over the course of development. This type of uncertainty is called epistemic uncertainty since it is reducible if greater knowledge is provided, see [9, 17, 20, 31].

Classical optimization methods seek an optimum in the design space. Typically, they do not consider the variability of design variables and do thus not take into account uncertainty. Consequently, optimal designs may be non-robust and quite sensitive to parameter variabilities, and are, therefore, infeasible. Some authors even believe that optimization is actually just the opposite of robustness, see [18]. As reliability is nevertheless mandatory, engineers have to look for robust designs which avoid unexpected deviations from the nominal performance, see [23]. To this end, more advanced methods have been developed to include uncertainties of the parameters and robustness criteria in the optimization.

Robust design optimization (RDO), as introduced in [32], includes robustness measures in the optimization problem. RDO helps to obtain a design that is less sensitive to variations of uncontrollable input variables without eliminating the source of the uncertainty, see [8, 10, 24, 32]. The solution of RDO is a robust design whose objective function value is insensitive to uncertainty, see [2].

Reliability-based design optimization (RBDO) is a method to determine optimal designs which are characterized by a low probability of failure. Prominent examples of RBDO are, for example, the first and second order reliability method (FORM/SORM), see [34]. It is assumed that the complete information of the input uncertainty is known, see [8, 19, 29]. This means, if there exists an inherent randomness in the non-deterministic behavior of the physical system, i.e., an aleatoric uncertainty, then this uncertainty is known and can be described, see [17, 20, 31].

Sensitivity analysis (SA) provides a further approach to deal with uncertainty. It aims at estimating the variability of the objective function value, affected by the variability of the input parameters, see [26]. Sensitivity measures for the input parameters are typically obtained by the ANOVA (analysis of variance) decomposition or by the Sobol decomposition, see [21, 28].

Uncertainty also arises when more than one design team is involved in the design of an engineering development process and every design team must optimize their subsystem without full information about the other subsystems. Every team has its own individual subsystem with goals and constraints which must match the goal of the overall design. Furthermore, the different disciplines (in e.g. vehicle crash development this are vibration analysis, durability, aerodynamics, etc.) may have conflicting objectives and the subsystems are often coupled, see [1, 14]. A possible

Key words and phrases. Robust design, uncertainty, solution space, high-dimensional systems, nonlinear systems.

method to solve such problems is *multidisciplinary design optimization (MDO)* because different disciplines are simultaneously optimized in MDO, see [6, 12].

Unfortunately, MDO, RBDO, RDO and SA suffer from certain disadvantages which excludes them for the application to the problems we have in mind. For MDO, a model which comprises all relevant disciplines must be provided. Unfortunately, such a model is usually not available for the design of complex engineering systems where different teams are involved in the development process. RBDO and RDO deal with models where the variability of input parameters is given. If, however, the variability of the input parameters is not known completely, it has to be estimated which is not always possible. When applying SA, information on how to improve a non-robust or critical solution is limited: what parameter needs to be adjusted and what value it should assume is unknown.

The strategy we follow here is to identify a maximum solution space for the high-dimensional, nonlinear, possibly noisy system under consideration [35]. On the solution space, the objective function assumes only subcritical output values, i.e., they are below a given threshold value, with a predefined probability for all enclosed designs. The solution space is expressed by intervals for each input parameter, representing a high-dimensional hyperbox. For a design to be good, the choice of a parameter value within its assigned interval does not depend on the values of the other parameters as long as they are within their respective intervals. In this sense, the parameters are decoupled from each other. The intervals may be used to assess robustness and sensitivity to uncertain input parameters which can be measured by the widths of the associated intervals. Moreover, a hyperbox helps to identify relevant parameters to improve a non-robust or insufficient design. Additionally, they might be combined with intervals of other disciplines – their cross sections are global solution spaces.

This approach is similar to RBDO, as threshold values for the objective function are used to distinguish between good and bad designs. It is similar to RDO in that input variations are considered. It differs from both, however, in that the degrees of freedom of the optimization problem are the permissible tolerance regions rather than the design variables themselves.

An iterative algorithm, consisting of two phases, is analyzed in this paper for the iterative identification of the hyperbox described above. The algorithm was introduced and compared with data mining and machine learning techniques in [35]. The starting point is a candidate hyperbox which is built around a design with subcritical objective function value. Then, this candidate hyperbox is iteratively evaluated and modified. In the first phase, called the *exploration phase*, the landscape of the optimization problem under consideration is explored. This phase consists of three steps. In the first step, a design of experiments is performed (e.g. by Monte Carlo sampling, see [27]). In the second step, a subset is identified which contains only good designs of the original design space. In the third step, the hyperbox is moved through the design space in order to find the hyperbox with maximum volume. The first phase is iterated until the hyperbox does not move any more. Then, the second phase starts, called the *consolidation phase*. It consists of a repeated application of the first two steps of the first phase until a hyperbox is identified which contains only subcritical outputs with a predefined probability. This paper focuses on the analysis of the algorithm's performance.

The paper is organized as follows. In Section 2, motivated by a problem from engineering practice, the problem statement is derived for identifying the maximum hyperbox which guarantees a subcritical performance. Section 3 introduces the algorithm which identifies hyperboxes. Three benchmark problems are given in Section 4 to validate the proposed algorithm by comparing the numerical results with the exact solutions. In addition, an engineering problem from vehicle front crash design confirms the applicability to high-dimensional and nonlinear engineering models. To analyze the reliability of the algorithm, the consolidation phase is studied in Section 5. This includes a study of the convergence behavior. In addition, the relevant mechanisms which are related to the problem's dimensionality are identified. With these results, the optimal number of sample points per iteration can be chosen, depending on the preference for speed or volume size. Finally, in Section 6, we state concluding remarks.

2. Motivation and problem statement

2.1. **Example problem.** We shall consider a model of a full-width front impact crash. The vehicle crashes head-on into a rigid concrete barrier at 56 km/h, see Figure 1(a). In the vehicle development, the maximum of the crash pulse generated by the vehicle structure is a relevant parameter to minimize the injury of car occupants in a front crash, see [33]. The crash pulse is the deceleration time history measured at the rocker panel and the B-pillar of the vehicle, see [11]. The crash pulse is entirely determined by the force-deformation characteristics of the elements of the car structure, parametrized by F_i, see [13]. A visualization of the force-deformation curve of a part of the front structure is shown in Figure 1(b) and Figure 1(c). Crash simulations show an inherently nonlinear physical behavior with respect to structural parameters. For this reason, the crash pulse is difficult to design.

An important objective quantity for crash design is the maximum of the crash pulse and is denoted as a. An optimization could be run for the function $a = f(F_1, F_2, \ldots, F_d)$ in order to find an optimum for the maximum of the crash pulse. Unfortunately, this solution cannot be realized exactly due to uncertainties. Therefore, rather than computing an optimum, a range of solutions, expressed as a hyperbox, should be computed. This hyperbox would represent admissible intervals for the degrees of freedom F_i.

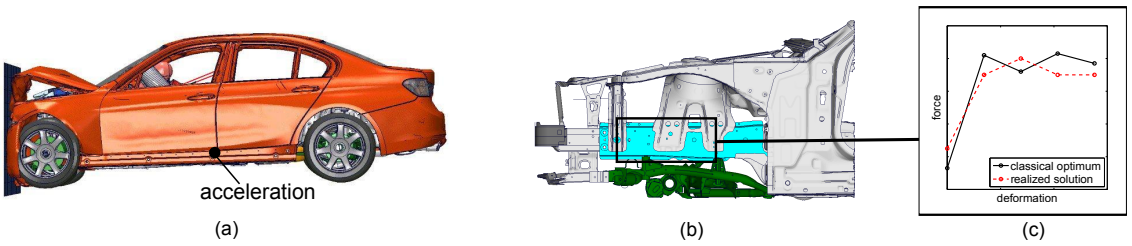


Figure 1. (a) Simulation of a vehicle front crash, (b) vehicle front structure and (c) force-deformation characteristics of a structural component of a front car structure.

2.2. **Problem statement.** Let $\Omega_{DS} \subset \mathbb{R}^d$ be a bounded, closed and convex set of admissible designs, called the design space. The volume of an axis-parallel hyperbox in this design space should be maximized under the constraint that the objective values of all enclosed designs are below a given threshold. The hyperbox corresponds to a product of intervals for each input parameter.

Definition 2.1. *Consider* $\mathbf{x}^{low} = (x_1^{low}, x_2^{low}, \ldots, x_d^{low})$, $\mathbf{x}^{up} = (x_1^{up}, x_2^{up}, \ldots, x_d^{up}) \subset \Omega_{DS}$ *such that* $\mathbf{x}^{low} \leq \mathbf{x}^{up}$ *component-by-component. Then, the hyperbox* $\Omega = \Omega(\mathbf{x}^{low}, \mathbf{x}^{up})$ *is the cartesian product*

$$\Omega = I_1 \times \cdots \times I_d \subset \Omega_{DS}$$

of intervals

$$I_i = [x_i^{low}, x_i^{up}] \subset \mathbb{R} \text{ for all } i = 1, 2, \ldots, d.$$

If we denote the width of the i-th interval I_i by $\ell_i = x_i^{up} - x_i^{low}$, then $\ell_1, \ell_2, \ldots, \ell_d$ are the lengths of the edges of the hyperbox Ω. Especially, $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$ is given by $\boldsymbol{\ell} = \mathbf{x}^{up} - \mathbf{x}^{low}$. The volume $\mu(\Omega)$ of the hyperbox Ω is thus given by

$$\mu(\Omega) = \prod_{i=1}^{d} \ell_i.$$

Let $f : \Omega_{DS} \to \mathbb{R}$ be an objective function which denotes a scalar quantity of interest. In practical applications, it represents a numerical simulation which produces a result $f(\mathbf{x})$ from input parameters $\mathbf{x}$. For the system $f(\mathbf{x})$ and a given critical value $f_c \in \mathbb{R}$, a hyperbox Ω is sought such that $\mu(\Omega) \to \max$ subject to $f(\mathbf{x}) \leq f_c$ for all $\mathbf{x} \in \Omega$.

Definition 2.2. *A design $\mathbf{x} \in \Omega_{DS}$ which satisfies the constraint $f(\mathbf{x}) \leq f_c$ is called a good design. A design $\mathbf{x} \in \Omega_{DS}$ which violates the constraint $f(\mathbf{x}) \leq f_c$ is called a bad design.*

With these preparations at hand, we can state the following constrained, nonlinear, and high-dimensional optimization problem:

$$\text{(P)} \qquad \left.\begin{array}{c} \text{find } \mathbf{x}^{low}, \mathbf{x}^{up} \in \Omega_{DS} \text{ with } \mathbf{x}^{low} \leq \mathbf{x}^{up} \text{ component-by-component} \\ \text{such that } \mu(\Omega) \to \max \text{ subject to } f(\mathbf{x}) \leq f_c \text{ for all } \mathbf{x} \in \Omega. \end{array}\right\}$$

This optimization problem is a shape optimization problem which shall be solved without the use of gradients to ensure applicability to any engineering problem where the function $f(\mathbf{x})$ is not analytically given. The solution will be a hyperbox which induces a fixed interval for each input parameter. In practice, these intervals define requirements for the related components and will be used in the development process as design goals.

Remark 2.3. *Typically, engineering problems have several design goals $f^{(i)}(\mathbf{x}) \leq f_c^{(i)}$, $i = 1, \ldots, m$. They are included in the problem formulation* (P) *by setting $f(\mathbf{x}) := \max_{i=1}^{m} \{f^{(i)}(\mathbf{x}) - f_c^{(i)}\}$ and $f_c := 0$.*

2.3. **Solution strategies.** There are already approaches in the literature to solve optimization problems similar to (P). In [25], this problem is solved by using an approach which combines cellular evolutionary strategies and interval arithmetic. Interval arithmetic is applied to evaluate the generated hyperbox. The drawback is that the objective function $f(\mathbf{x})$ has to be known analytically and can thus not be treated as a black box (see [5]). Unfortunately, in most practical applications from engineering, the objective function is represented by a numerical simulation, producing a result $f(\mathbf{x})$ from input parameters $\mathbf{x}$, and is not known analytically.

In other approaches, admissible design domains are identified with the aid of cluster analysis and fuzzy set theory (see [3, 22]). Nevertheless, fuzzy set theory needs additional information like the membership function of the parameters which is typically not available in the engineering design development. Furthermore, the design space is sampled only once. Consequently, the number of sample points has to be larger than the volume of the design space divided by the volume of the solution space to detect good regions. For high-dimensional problems with many relevant input parameters, a very large number of sample points is required to identify the solution space.

Surrogate modelling can be used for the fast evaluation of a design point $\mathbf{x}$ (for mathematical surrogates see [5], for an example of physical surrogate modelling see [7]). It can be combined with the approach presented here, however, it cannot replace it, as their output are performance values of single design points rather than solution spaces.

The algorithm which is considered in this paper has been introduced in [35]. It aims at directly solving the constrained nonlinear high-dimensional optimization problem (P). This algorithm requires only function evaluations and, therefore, no access to the analytical expression of $f(\mathbf{x})$ which means the system $f(\mathbf{x})$ is treated as a black box. Thus, the proposed optimization method is non-intrusive.

3. Algorithm

The algorithm for solving (P) consists of two phases as seen in the flowchart in Figure 2. The starting point is a design $\mathbf{x}$ which fulfills the inequality $f(\mathbf{x}) \leq f_c$. It can be found by a classical optimization like differential evolution (see e.g. [30]). Then, an initial hyperbox is generated which includes this admissible design. The size of the initial box has to be chosen sufficiently small in order to ensure that by sampling at least one good design is found. The size may be highly dependent on the problem and may has to be adapted accordingly. For all results presented in this paper, the edge length of the initial box in each dimension has been set to be 20% of the respective design interval with the admissible design as center. The first phase, called the *exploration phase*, is an iterative scheme which explores the landscape of the objective function. Finally, the second phase of the algorithm, called the *consolidation phase*, is performed. The consolidation phase includes an algorithm which shrinks the hyperbox such that it contains only good designs at a given probability level. The total number function calls which have to be calculated to identify

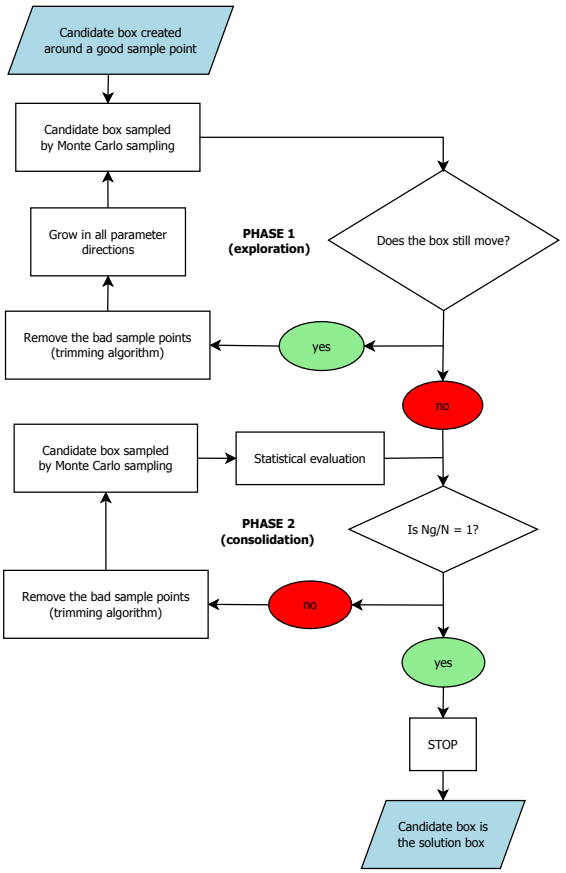


FIGURE 2. The flowchart of the algorithm to identify the maximal hyperbox.

the largest hyperbox is $N \times M$, where N is the number of sample points per iteration and M is the total number of iteration steps in the exploration phase and in the consolidation phase.

3.1. **Exploration phase.** The exploration phase consists of three basic steps which are outlined in the flowchart in Figure 2 at the top.

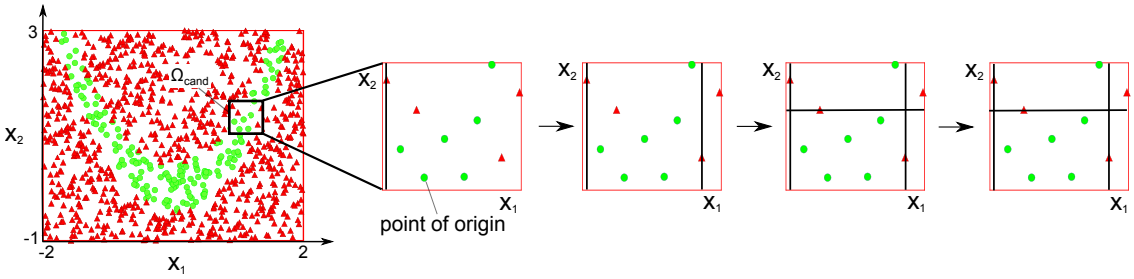


FIGURE 3. Trimming algorithm to select the hyperbox with the most good sample points in Ω.

In the first step, i.e., the hyperbox evaluation, a population of designs is created by using a design of experiments technique such as Monte Carlo sampling or Latin hypercube sampling (see e.g. [27]) in the candidate hyperbox $\Omega = [x_1^{low}, x_1^{up}] \times \cdots \times [x_d^{low}, x_d^{up}]$. The generated population $\{\mathbf{x}_j\}$ is divided in good sample points which fulfill $f(\mathbf{x}_j) \leq f_c$ and bad sample points for which it holds $f(\mathbf{x}_j) > f_c$.

Definition 3.1. *The ratio $\widehat{a} = N_g/N$ of the number N_g of good sample points and the total number N of sample points is called the fraction of good sample points.*

In the second step, the hyperbox is modified by removing all bad sample points. This is done by an algorithm which identifies a hyperbox that includes only good sample points according to the following rule.

A good sample point is used as the point of origin, i.e., it will always be included, as visualized in Figure 3. Then, the bad sample point with the highest objective value is removed by relocating the boundaries such that the smallest number of good sample points is lost. If this dimension is not unique, i.e., if there is more than one dimension associated with the loss of the same number of good sample points, then that one is chosen where most bad sample points are removed. The next bad sample point with the remaining highest value is removed in such a way that again the smallest number of good sample points are lost as shown in Figure 3. This procedure is repeated until there are no more bad sample points. When all bad sample points are removed, the hyperbox is again reduced in size with respect to all dimensions where a bad sample point was removed such that the new box boundaries will lie on good sample points.

After this process, for each good sample point, there will be a hyperbox which contains only good sample points. From all these hyperboxes, the one with the most sample points will be selected as the output.

Remark 3.2. *The algorithm will be called trimming algorithm as it removes the bad space by relocating the boundaries.*

The pseudo-code of the trimming algorithm is found in Algorithm 1. Its computational complexity is $\mathcal{O}(N^3 d)$ where N is the total number of sample points and d is the number of dimensions.

Algorithm 1: Pseudo-code of the trimming algorithm.

Data: a hyperbox Ω and a set $\mathcal{S} = \{\mathbf{x}_j \in \Omega : f(\mathbf{x}_1) \geq \cdots \geq f(\mathbf{x}_N)\}$ of sample points
Result: hyperbox $\subseteq \Omega$ which contains only good sample points
forall the *good sample points* $\{\mathbf{x}^{good} \in \mathcal{S} : f(\mathbf{x}^{good}) \leq f_c\}$ **do**
  forall the *bad sample points* $\{\mathbf{x}^{bad} \in \mathcal{S} : f(\mathbf{x}^{bad}) > f_c\}$ **do**
    for $i = 1, 2, \ldots, d$ **do**
      if $x_i^{bad} < x_i^{good}$ **then**
        count the good sample points $\mathbf{x}$ with $x_i^{bad} \geq x_i \geq x_i^{low}$;
      else
        count the good sample points $\mathbf{x}$ with $x_i^{bad} \leq x_i \leq x_i^{up}$;
    choose the direction $i^\star$ where the fewest good sample points are removed;
    if $x_{i^\star}^{bad} < x_{i^\star}^{good}$ **then** trim to $x_{i^\star}^{low}$;
    else trim to $x_{i^\star}^{up}$;
  forall the *directions i where a bad sample point is removed* **do**
    if $x_i^{bad} < x_i^{good}$ **then**
      $x_i^{low} := \min_j x_{i,j}$ for all remaining good sample points $\mathbf{x}_j$;
    else
      $x_i^{up} := \max_j x_{i,j}$ for all remaining good sample points $\mathbf{x}_j$;
  remember the hyperbox with most good sample points;

The third step consists of modifying the candidate hyperbox by growing in all parameter directions. By increasing the size of the hyperbox, new regions are enclosed which enable the hyperbox to move through the design space in order to find the best hyperbox. The hyperbox is enlarged by adjusting its boundaries as follows. The boundaries in the $(k+1)$-st iteration are calculated from the boundaries of the k-th iteration according to

$$[x_i^{low}]_{k+1} = [x_i^{low}]_k - \beta_k\big([x_i^{up}]_k - [x_i^{low}]_k\big) \text{ for all } i = 1, 2, \ldots, d$$

for the lower boundary and according to

$$[x_i^{up}]_{k+1} = [x_i^{up}]_k + \beta_k\big([x_i^{up}]_k - [x_i^{low}]_k\big) \text{ for all } i = 1, 2, \ldots, d$$

for the upper boundary. If the growth rate β_k is chosen too large, then the fraction of good sample points in the next candidate hyperbox will be small. A small number of good sample points provides only little information about further boundary relocations, thus making over- and underestimation more likely (see Section 5). If the growth rate β_k is chosen too small, many iteration steps will be necessary in the exploration phase. As a compromise between speed and accuracy, the growth rate is adaptively chosen to be

$$\beta_k = \frac{\widehat{a}_{k-1}}{a_{target}}\beta_{k-1}, \tag{1}$$

where k is the iteration index, $\widehat{a}_{k-1}$ is the fraction of good sample points in the $(k-1)$-st iteration, and a_{target} is the desired fraction of good sample points. To our experience, the choice $a_{target} = 0.8$ leads to good results in (1).

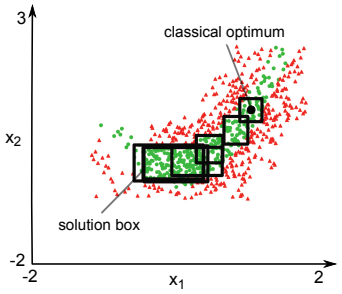


FIGURE 4. Iteration process starting in the classical optimum to move towards a beneficial direction with increasing box size.

The complete exploration phase is illustrated in Figure 4 in case of the Rosenbrock function

$$f(x_1, x_2) = (1 - x_1)^2 + 100\big(x_2 - x_1^2\big)^2.$$

The design space is $\Omega_{DS} = [-2, 2] \times [-2, 3]$ and the critical value is $f_c = 20$. The initial β-value was chosen to be 1. This value is small enough to obtain a sufficiently high fraction of good designs in the first iteration step. It is also large enough to ensure that not too much iteration steps are performed in the exploration phase. It can be observed that the hyperbox moves away from the classical optimum towards a beneficial direction with larger box size. If the hyperbox does not move and $\mu(\Omega)$ does not significantly change anymore, the algorithm switches to the second phase.

3.2. **Consolidation phase.** The consolidation phase is an iterative scheme where each iteration consists of three basic steps as seen in the flowchart in Figure 2. In the first step, bad sample points are removed by the trimming algorithm as described in the previous section. In the next step, again, a sample of designs is created by using Monte Carlo sampling or Latin hypercube sampling in the candidate hyperbox. Then, the candidate hyperbox is statistically evaluated in the third step. Given the lower and upper boundary of a Bayesian confidence interval $[a_{low}, a_{up}]$ for the probability of success a, the probability for a to lie within that confidence interval is calculated according to the following theorem (see [16] for a quantitative study of this result).

Theorem 3.3 ([16]). *Let $N \in \mathbb{N}$ be the total number of uniformly distributed sample points in the hyperbox Ω and $N_g \leq N$ the number of good sample points. Moreover, let $a \in [0, 1]$ denote the true fraction of the good space in the hyperbox Ω. Then, the confidence level that the probability of the fraction of good sample points (probability of success) lies within the Bayesian confidence interval $[a_{low}, a_{up}]$ is*

$$P(a_{low} < a < a_{up}|N_g) = \frac{\int_{a_{low}}^{a_{up}} t^{N_g}(1-t)^{N-N_g}\mathrm{d}t}{\int_0^1 s^{N_g}(1-s)^{N-N_g}\mathrm{d}s}. \tag{2}$$

In Figure 5, the width of the Bayesian confidence interval around the probability of success is shown over the number of sample points N for different values of N_g/N. As it can be observed, the width of the confidence interval decreases when the ratio N_g/N increases. When there are 100 sample points and the ratio $N_g/N = 1$, which means that there are only good sample points, the true fraction of the good space is with 95% probability between $a_{low} = 0.97$ and $a_{up} = 1$, see [16]. Therefore, the width of the 95%-confidence interval is 0.03. The remaining 3% probability of failure can be accepted for two reasons. First, the determination of a solution hyperbox is useful in the early development stage in order to design a product which fulfills the requirements. The layout of, e.g., a vehicle is likely to change over the course of the development process. There is thus no need for a high accuracy. Second, the final design will always be verified by a detailed simulation or a hardware test to confirm the fulfillment of the requirements. The width of 0.03 of the 95%-confidence interval is assumed to be sufficiently small, especially in an early development stage where the knowledge of the final design is limited. Therefore, in general $N = 100$ sample points are used for the evaluation of the candidate hyperbox.

When $N_g/N = 1$, the algorithm will stop and the last candidate hyperbox is chosen to be the final solution hyperbox.

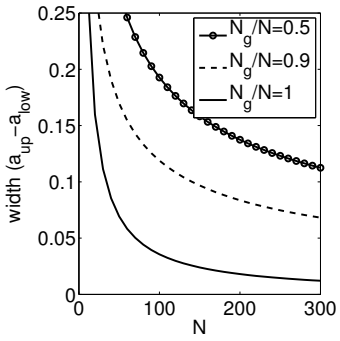


FIGURE 5. The width of the 95%-confidence interval around the probability of success.

4. Results of the algorithm

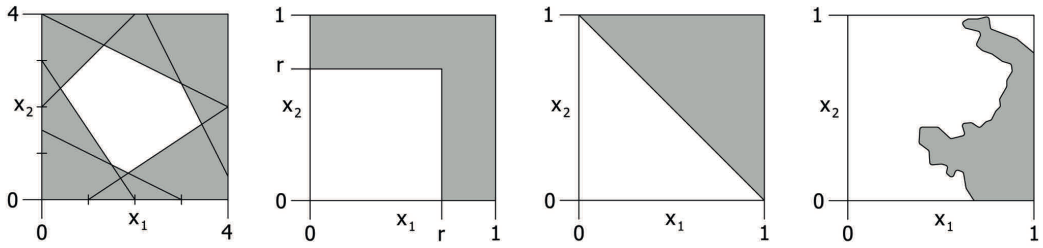
In this section, three benchmark problems are considered to compare the solution of the proposed algorithm with the exact solution of the optimization problem (P). This means that the *accuracy* of the algorithm is studied. To that end, always a sufficiently large, fixed number of iterations is chosen in both, the exploration phase and the consolidation phase. Moreover, also a fixed number of 100 sample points is used in each iteration which turned out to be a good choice according to the results of Problem 1 and Section 5.4. Note that the algorithm's *speed of convergence* will be investigated in Section 5.

- In Problem 1, the solution space is constrained by a convex polytope in two dimensions as depicted in Figure 6(a).
- In Problem 2, the good space is given by a hyperbox which is inscribed in the design space (see Figure 6(b)). The ratio of the volume of the good space and the volume of the design space is 0.5.
- In Problem 3, the solution space is constrained by a tilted hyperplane which divides the d-dimensional unit cube into two equal volumes, cf. Figure 6(c).

These problems are nonlinear optimization problems under affine constraints. The corresponding total performance function is nonlinear for Problems 1 and 2. High-dimensionality is considered in Problems 2 and 3.

In addition, the applicability of the solution algorithm to high-dimensional and nonlinear engineering problems is demonstrated by means of Problem 4 from crash analysis which has been

described in Subsection 2.1. The gray surface in Figure 6(d) illustrates the bad space of a particular two-dimensional cross section.



(a) (b) (c) (d)

FIGURE 6. Problems considered: (a) Problem 1 (convex polytope), (b) Problem 2 (hyperbox), (c) Problem 3 (tilted hyperplane), and (d) Problem 4 (front crash).

4.1. **Problem 1. A convex polytope as boundary.** As an illustrative example, in particular, for validating the algorithm's convergence towards the correct solution, we consider the nonlinear optimization problem of maximizing a rectangle within a convex polytope in two dimensions (cf. Figure 6(a)).

4.1.1. *Exact solution.* Let $\Omega_{DS} = [0,4]^2$ be the design space and

$$\Omega = [x_1^{up} - x_1^{low}] \times [x_2^{up} - x_2^{low}] \subset \Omega_{DS}.$$

Writing $\mathbf{z} = (\mathbf{x}^{low}, \mathbf{x}^{up}) = (x_1^{low}, x_2^{low}, x_1^{up}, x_2^{up})$, the set

$$K = \{(\mathbf{x}^{low}, \mathbf{x}^{up}) \in \Omega_{DS} \times \Omega_{DS} : \mathbf{g}(\mathbf{z}) = \mathbf{A}\mathbf{z} - \mathbf{b} \leq \mathbf{0} \text{ and } \mathbf{x}^{low} \leq \mathbf{x}^{up}\},$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \frac{1}{8} & \frac{1}{4} \\ 0 & 0 & \frac{4}{17} & \frac{2}{17} \\ -\frac{1}{2} & 0 & 0 & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & -\frac{2}{3} & 0 & 0 \\ 0 & -\frac{3}{2} & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix},$$

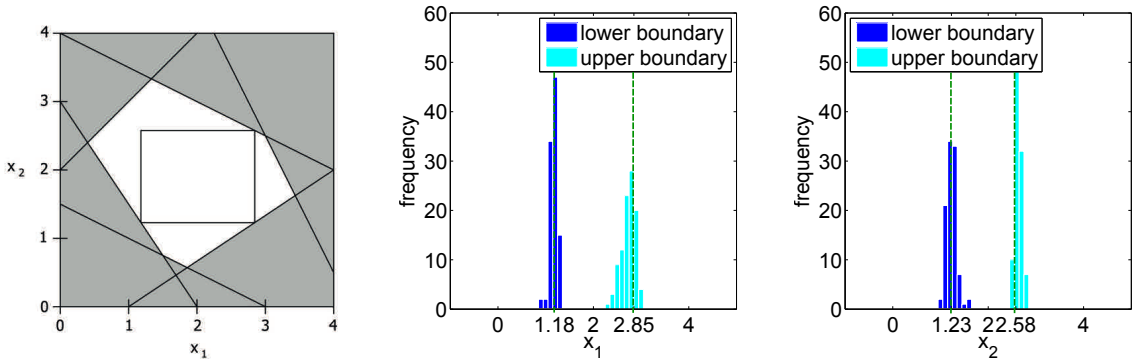
describes the convex polytope seen in Figure 7(a). Due to

$$\mu(\Omega) = (x_1^{up} - x_1^{low})(x_2^{up} - x_2^{low}) = \frac{1}{2}\mathbf{z}^T\mathbf{D}\mathbf{z} \quad \text{with} \quad \mathbf{D} = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 2 \\ -2 & 0 & 0 & 0 \end{bmatrix},$$

the constrained optimization problem

$$\mu(\Omega) \to \max \text{ subject to } \mathbf{z} \in K \tag{3}$$

is a quadratic optimization problem under affine inequality constraints. The unique maximum $\mu(\Omega)$ is found by means of Lagrange multipliers (see e.g. [15]) and has the values tabulated in the second column of Table 1. A visualization of this maximum is found in Figure 7(a).



(a) (b) (c)

FIGURE 7. Problem 1. (a) Convex polytope in $d = 2$ dimensions with the hyperbox of maximal volume. (b) Distribution of the hyperboxes found by the algorithm for x_1 in 100 simulations. (c) Distribution of the hyperboxes found by the algorithm for x_2 in 100 simulations.

4.1.2. *Numerical solution.* The constrained optimization problem (3) can equivalently be expressed as the constrained optimization problem (P) if we set

$$f(x_1, x_2) = \begin{cases} 0, & \text{if } \mathbf{g}(x_1, x_2, x_1, x_2) \leq \mathbf{0} \text{ component-by-component,} \\ 1, & \text{otherwise,} \end{cases}$$

and $f_c = 0.5$. Therefore, problem (3) can be solved numerically by using the algorithm presented in Section 3. The results of the numerical optimization are shown in Table 1 for $N = 50$ and $N = 100$ sample points per iteration. The algorithm was run 100 times, where the iterative process was started with a hyperbox built around a randomly chosen admissible point. 200 iterations of the exploration phase and 100 iterations of the consolidation phase were used. The columns entitled $x_{i,avg}$ contain the mean of the coordinates of the final hyperboxes of the 100 simulations, the columns entitled $\sigma(x_i)$ are the related standard deviations, the columns entitled $\varepsilon(x_i)$ contain the absolute errors $|x_{i,avg} - x_{i,opt}|$, and the columns entitled "error" contain the relative errors $|x_{i,avg} - x_{i,opt}|/x_{i,opt}$.

	exact	numerical solution ($N = 50$ samples)				numerical solution ($N = 100$ samples)			
	$x_{i,opt}$	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %
x_1^{low}	1.18	1.181	0.101	0.001	0.058	1.174	0.074	0.006	0.508
x_1^{up}	2.85	2.687	0.169	0.163	5.708	2.735	0.140	0.115	4.050
x_2^{low}	1.23	1.245	0.138	0.015	1.203	1.235	0.108	0.005	0.407
x_2^{up}	2.58	2.647	0.092	0.067	2.589	2.633	0.073	0.053	2.052
average	—	—	0.125	0.062	2.390	—	0.099	0.045	1.754

TABLE 1. Exact solution of Problem 1 and related numerical results for 100 simulations with $N = 50$ and $N = 100$ samples per iteration.

We observe in Table 1 that the standard deviation decreases if the number of sample points per iteration is doubled. The mean error is 2.4% for $N = 50$ and 1.8% for $N = 100$. Therefore, the error between the exact solution and the mean of the numerical solutions becomes smaller when the number of sample points per iteration is increased. In Section 5.4, it is shown that $N = 100$ sample points are a good choice to converge fast and to obtain a large volume in the consolidation phase of the algorithm. Therefore, $N = 100$ is chosen in the next test examples.

The distribution of the solution hyperboxes found by the algorithm is visualized for $N = 100$ sample points in Figure 7(b) for the coordinate x_1 and in Figure 7(c) for the coordinate x_2. The

boundaries of the optimum hyperbox are indicated by the dashed green lines. These plots show that the proposed algorithm converges to the exact solution in the sense that the average of the numerical solutions approximately coincides with the exact solution.

4.2. **Problem 2. A hyperbox as boundary.** The maximization of a hyperbox in case of another hyperbox being the boundary between the good and the bad space is considered in $d > 1$ dimensions. In particular, the convergence of the algorithm to the optimal solution is shown in high dimensions.

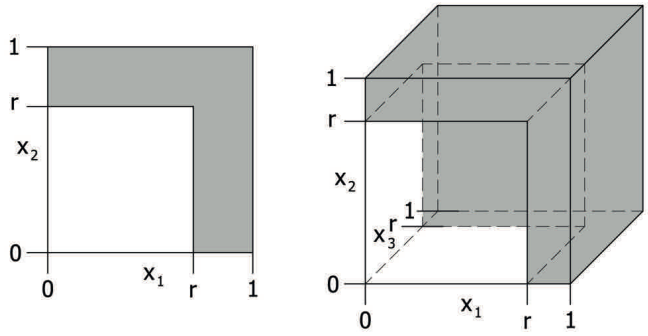


FIGURE 8. Problem 2. A hyperbox which defines the good space of the design space, inscribed in the unit d-cube.

4.2.1. *Exact solution.* For $r \leq 1$, let $[0,r]^d \subset \mathbb{R}^d$ be a hyperbox which defines the good space of the design space. This hyperbox is inscribed in the d-dimensional unit cube $\Omega_{DS} = [0,1]^d$ which serves as the design space. The value of $r = r(d)$ is chosen in such a way that the fraction of the good space is always 0.5, that is $r = \sqrt[d]{0.5}$ (see Figure 8 for a visualization in the case of $d = 2$ and $d = 3$ dimensions). Hence, with the lower bounds $x_i^{low} = 0$ for all $i = 1, 2, \ldots, d$, we arrive for $x_i = x_i^{up}$, $i = 1, 2, \ldots, d$, at the optimization problem

$$\mu(\Omega) = \prod_{i=1}^{d} x_i \to \max \tag{4}$$

under the affine inequality constraints

$$0 \leq x_i \leq r \text{ for all } i = 1, 2 \ldots, d. \tag{5}$$

The solution to this optimization problem is easily calculated by $x_i = x_{opt} = \sqrt[d]{0.5}$ for all $i = 1, 2 \ldots, d$. The value x_{opt} tends to 1 as d tends to infinity which is also seen by dashed blue curve in Figure 9.

4.2.2. *Numerical solution.* The solution of the optimization problem (4) under the inequality constraint (5) coincides with the solution of problem (P) with

$$f(\mathbf{x}) = \begin{cases} 0, & \text{if } x_i \leq r \text{ for all } i = 1, 2, \ldots, d, \\ 1, & \text{otherwise,} \end{cases}$$

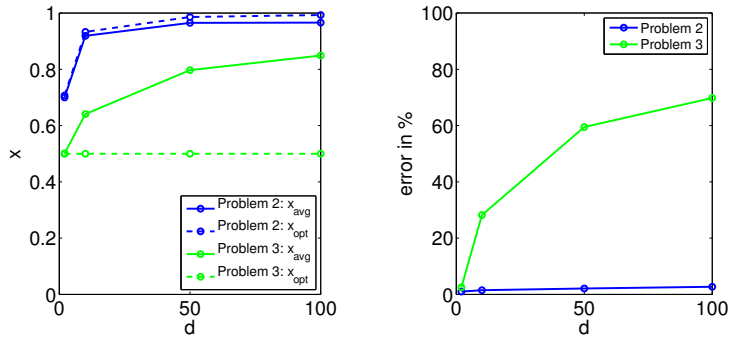
and $f_c = 0.5$. This optimization problem is solved numerically for $d = 2, 3, 10, 50, 100$ spatial dimensions. The algorithm is run 100 times with $N = 100$ sample points per iteration. The iterative process starts with a hyperbox built around a randomly chosen admissible point.

In Table 2, the mean of the numerical solutions and the standard deviations are tabulated for $d = 2$ and $d = 3$, respectively. Herein, $x_{i,avg}$ denotes the mean of the i-th coordinate of the final hyperbox. The associated standard deviation is $\sigma(x_i)$, the absolute error is $\varepsilon(x_i) = |x_{i,avg} - x_{i,opt}|$, and the relative error is $|x_{i,avg} - x_{i,opt}|/x_{i,opt}$. The agreement between the numerical solutions of the algorithm and the exact solutions is reasonably good.

$d=2$	exact	numerical solution			
	$x_{i,opt}$	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %
x_1	0.707	0.700	0.007	0.007	0.95
x_2	0.707	0.700	0.007	0.007	1.05
$d=3$	exact	numerical solution			
	$x_{i,opt}$	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %
x_1	0.794	0.784	0.010	0.010	1.20
x_2	0.794	0.784	0.009	0.010	1.23
x_3	0.794	0.784	0.010	0.010	1.20

TABLE 2. Exact solution of Problem 2 and related numerical results for 100 simulations in $d=2$ and $d=3$ dimensions.

To evaluate the results of the algorithm when the number of dimensions increases, the boundaries of the resulting hyperboxes $x_{i,\ell}$ are computed for every run ℓ. The mean x_{avg} with respect to all (i,ℓ) is presented in Figure 9(a) for $d=2,10,50,100$ dimensions. One infers that the mean x_{avg} approximately agrees with the analytical mean $x_{opt}=\sqrt[d]{0.5}$, also in high dimensions. The relative error $|x_{avg}-x_{opt}|/x_{opt}$ is plotted in Figure 9(b). It is nearly independent of the number of dimensions, and it is smaller than 3%. Therefore, the algorithm approximates the optimum hyperbox well also in high dimensions.



(a) (b)

FIGURE 9. Numerical and exact solutions for Problem 2 (hyperbox) and Problem 3 (tilted hyperplane). (a) Mean of the averaged simulations in comparison with the optimal solutions for $d=2,10,50,100$. (b) The relative error for $d=2,10,50,100$.

4.3. **Problem 3. A tilted hyperplane as boundary.** We consider now a d-dimensional problem with a tilted hyperplane as boundary of the good space. The exact solution is computed and the algorithm's behavior for large d is studied.

4.3.1. *Exact solution.* Let us consider the d-dimensional unit cube $[0,1]^d$ as design space Ω_{DS}. The boundary of the good space is a diagonal hyperplane which contains the point $[1,1,\dots,1]/2$ and has the normal vector $[1,1,\dots,1]/\sqrt{d}$. That is, the tilted hyperplane is described by the equation

$$g(\mathbf{x})=\sum_{i=1}^{d}x_i-\frac{d}{2}$$

and intersects the design space in the middle, see Figure 10(a) for $d=2$ and Figure 10(b) for $d=3$. This choice ensures that the fraction of the good space is 50%, independently of the dimension d.

The associated optimization problem for the hyperbox with the lower bounds $x_i^{low} = 0$ and the upper bounds $x_i = x_i^{up}$, $i = 1, 2, \ldots, d$, reads as

$$\mu(\Omega) = \prod_{i=1}^{d} x_i \to \max \text{ subject to } \mathbf{x} \in K = \{\mathbf{x} \in \Omega_{DS} : g(\mathbf{x}) \leq 0\}. \tag{6}$$

Its unique solution is given by $\mathbf{x} = [1/2, 1/2, \ldots, 1/2]$, cf. Figure 10 for an illustration of the associated optimum hyperbox. The optimum hyperbox volume is $\mu(\Omega^{opt}) = 2^{-d}$.

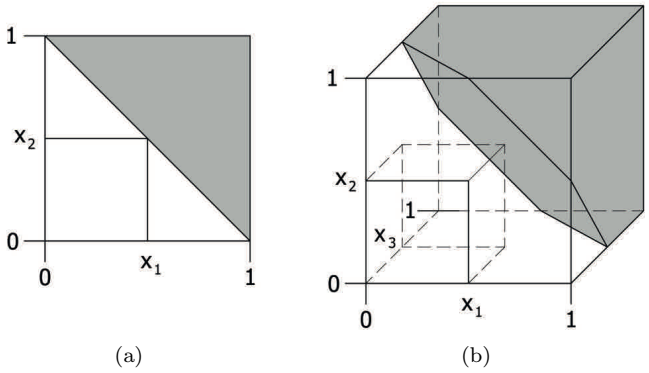


FIGURE 10. Problem 3. Tilted hyperplane in $d = 2$ and $d = 3$ dimensions with the maximal hyperbox.

4.3.2. *Numerical solution.* The constrained optimization problem (6) is equivalent to the constrained optimization problem (P) if we set $f(\mathbf{x}) = g(\mathbf{x})$ and $f_c = 0$. We shall compare the numerical results produced by the optimization algorithm with the exact solution. The algorithm is run 100 times where the initial guess is a hyperbox built around a randomly chosen admissible design.

For low dimensions, $d = 2$ and $d = 3$, the mean of the numerical solutions and the standard deviation are tabulated in Table 3. The mean of the i-th coordinate of the final hyperboxes $x_{i,avg}$, the associated standard deviation $\sigma(x_i)$, the absolute error $\varepsilon(x_i)$, and the relative error are calculated. These results confirm that the proposed algorithm approximates the exact solution in $d = 2$ and $d = 3$ dimensions.

$d = 2$	exact	numerical solution			
	$x_{i,opt}$	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %
x_1	0.5	0.479	0.06	0.021	4.2
x_2	0.5	0.523	0.06	0.023	4.6
$d = 3$	exact	numerical solution			
	$x_{i,opt}$	$x_{i,avg}$	$\sigma(x_i)$	$\varepsilon(x_i)$	error in %
x_1	0.5	0.515	0.08	0.015	3.0
x_2	0.5	0.508	0.07	0.008	1.6
x_3	0.5	0.519	0.07	0.019	3.8

TABLE 3. Exact solution of Problem 3 and related numerical results for 100 simulations for $d = 2$ and $d = 3$ dimensions.

Next, we compare the numerical result produced by the algorithm with the exact solution when the number of dimensions increases. We consider $d = 2, 10, 50, 100$ dimensions. As in the previous problem, the mean x_{avg} is computed with respect to all runs and all vertices. It is plotted in Figure 9(a) versus the spatial dimension. In low dimensions, the result of the algorithm is in good

agreement with the optimal solution. In high dimensions, that is $d > 10$, the algorithm strongly overestimates the optimum hyperbox. Hence, the associated relative error $|x_{avg} - x_{opt}|/x_{opt}$, found in Figure 9(b), strongly increases as the number of dimensions increases. Nevertheless, the fraction of bad space contained in the hyperbox is small and of the order $1/N$. This effect reflects the curse of dimensionality (see [4]), caused by the fact that the tilted hyperplane has no axis-parallel boundaries. This is in contrast to Problem 2 where the fraction of bad space contained in the hyperbox is very sensitive to overestimation since the boundaries between the good and the bad space are axis-parallel.

The algorithm ensures that the fraction of good designs N_g/N is sufficiently close to 1 in the sense of Theorem 3.3. Independently of the number of dimensions, the probability for a hyperbox to contain only good design points can be computed from sample data and can therefore be controlled. The volume of the resulting hyperbox may differ from the exact solution. This effect is particularly strong for problems with tilted solution space boundaries, i.e. the performance function depends on many parameters. A detailed analysis on how the number of samples influences the results is shown in Subsection 5.3.

4.4. **Problem 4. An engineering problem from front vehicle crash design.** A vehicle front crash problem, as described in Subsection 2.1, is considered to demonstrate the applicability to high-dimensional, nonlinear and noisy industrial problems. The front car structure of the vehicle consists of sixteen parts which are relevant for the crash design. The number of parameters (force levels) per part is four. In total, there are $d = 64$ parameters. The optimization problem under inequality constraints is given as follows:

$$\left.\begin{array}{c} \text{find } \mathbf{F}^{low}, \mathbf{F}^{up} \in \Omega_{DS} \text{ with } \mathbf{F}^{low} \leq \mathbf{F}^{up} \text{ component-by-component} \\ \text{such that } \mu(\Omega) \to \max \text{ subject to } a_{pulse} = f(\mathbf{F}) \leq f_c \text{ for all } \mathbf{F} \in \Omega. \end{array}\right\} \tag{7}$$

Here, the function $f : \Omega_{DS} \to \mathbb{R}$ is a mapping which is provided by a numerical simulation. The underlying computer model is described in [7] and consists of one-dimensional force elements which are connected by nodes that can only vary in the direction of the car's movement. The computation is performed by ABAQUS and takes less than 10 seconds on a single processor of a Linux workstation, having two Intel(R) Xeon(R) X5550 CPUs with a clock rate of 2.67 GHz and a main memory of 12 GB.

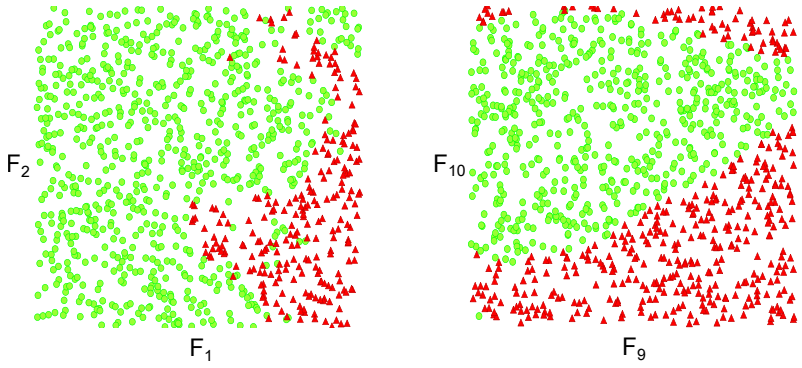


FIGURE 11. Problem 4. Two-dimensional cross sections of the design space. Shown are good (circles) and bad (triangles) design points.

Two-dimensional cross sections of the design space are shown in Figure 11 where circles define good designs which satisfy the constraint $f(\mathbf{F}) \leq f_c$ and triangles define bad designs which violate this constraint. Here, the highly nonlinear structure of the optimization problem (7) can be observed. Also, in the left diagram of Figure 11, regions can be observed where good and bad designs mix, suggesting erratic variations of the objective function. In crash simulations, this is referred to as *noisy* behavior.

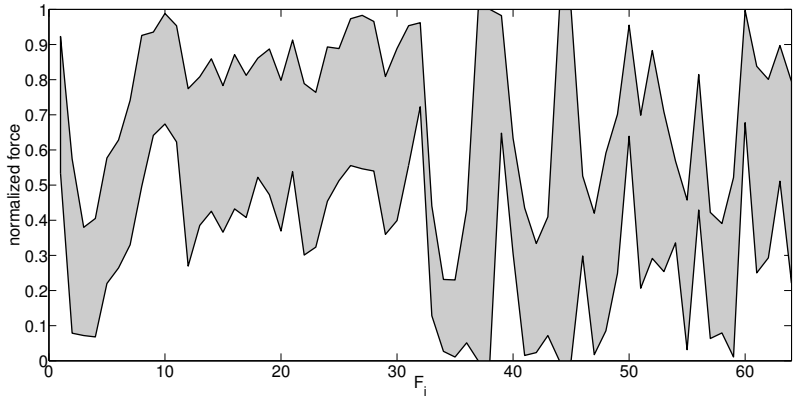


FIGURE 12. Normalized intervals for each parameter F_i.

The algorithm is applied to the optimization problem (7), where 200 iterations are performed in the exploration phase and 200 iterations are performed in the consolidation phase. Hence, the algorithm needs $N \times M = 100 \times 400$ function calls to compute the solution hyperbox. This hyperbox is illustrated in Figure 12 via normalized intervals $[F_i^{low}, F_i^{up}]$ for each parameter F_i with $i = 1, 2, \ldots, 64$.

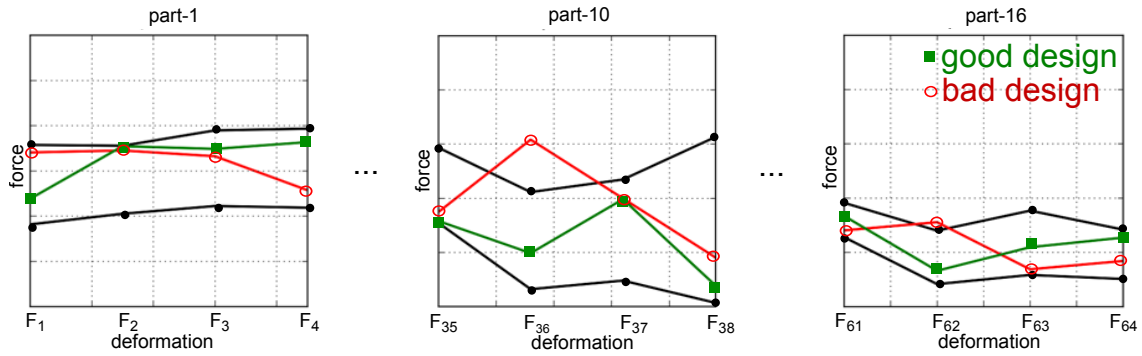


FIGURE 13. Force-deformation intervals for a vehicle structure. A bad and a good design are shown.

In Figure 13, the calculated force-deformation intervals are presented for three of the 16 parts of the front car structure. Particular good (green line) and bad (red line) designs are also shown in Figure 13. The bad design lies outside the computed hyperbox because the design is not in the given interval at part 10 and 16. Nevertheless, note that not all designs which are outside of the hyperbox have to be bad.

A sample with 100 arbitrarily chosen force-deformation curves which are inside the solution hyperbox are depicted in Figure 14 for three of the 16 parts of the front car structure. For all curves, the maximal crash pulse is indeed subcritical. Hence, N_g/N is equal to 1 and, according to Theorem 3.3, the true fraction of the good space is between 0.97 and 1 with 95% probability.

In Figure 15(a), the fraction of good sample points is shown for different phases of the algorithm. During the exploration phase (phase 1), the fraction of good sample points oscillates between 0.7 and 0.9. In the consolidation phase (phase 2), the curve converges towards 1 as desired. The normalized volume is found in Figure 15(b). It grows in the exploration phase of the algorithm and decreases in the consolidation phase. Here, $\mu(\Omega_{box})$ denotes the volume of the candidate hyperbox and $\mu(\Omega_{DS})$ is the volume of the design space. Figure 15(c) shows the normalized volume versus the fraction of good sample points. In the exploration phase, the curve stagnates, and converges then in the consolidation phase. The algorithm converges to a hyperbox with a fraction of good

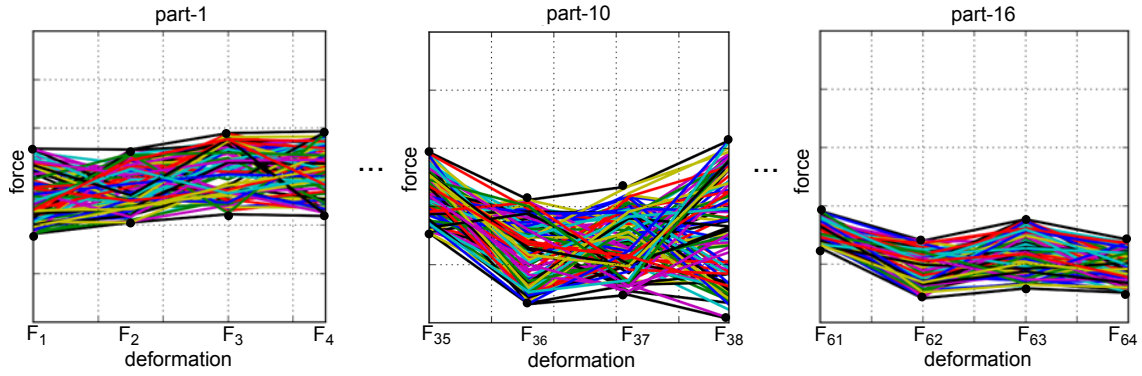
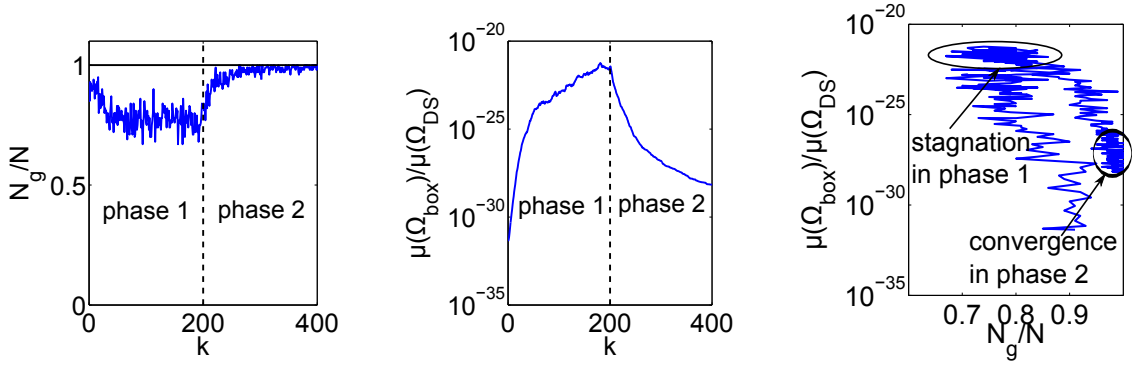


FIGURE 14. Force-deformation intervals in case of the front crash with $N = 100$ sample points.

space of 100% which is illustrated by normalized intervals in Figure 12. This validates that the algorithm is applicable to high-dimensional, nonlinear problems from engineering applications.



(a) (b) (c)

FIGURE 15. Problem 4. (a) The fraction of good sample points versus the number of iterations. (b) The hyperbox volume versus the number of iterations. (c) The hyperbox volume over the fraction of good sample points.

5. Convergence behavior in the consolidation phase

The user of the algorithm seeks a hyperbox with a large volume $\mu(\Omega)$. Furthermore, a large fraction N_g/N of the good design space is desired. This fraction is typically small during the exploration phase. The purpose of the consolidation phase is to increase this fraction to a required level, possibly at the cost of a smaller hyperbox volume. This section studies how the number of dimensions and the number of sample points affect the quality of the resulting solution hyperbox and the speed of convergence in the consolidation phase.

First, in Subsection 5.1, a convergence coefficient is introduced as a measure of the convergence speed. Then, in Subsection 5.2, Problems 2 (hyperbox), 3 (tilted hyperplane), and 4 (front crash) are considered to investigate the influence of the dimensionality on the convergence behavior. Afterwards, Subsection 5.3 studies the influence of the sample size on the speed of convergence and on the volume of a resulting solution hyperbox. Then, in Subsection 5.4, it is demonstrated that the speed of convergence and the volume of the resulting solution hyperbox are typically in conflict with each other. This conflict is illustrated by appropriate Pareto lines for all problem under consideration. Depending on the preference for speed or volume size, the sample size can be chosen and the total number of required simulations can be estimated.

5.1. **Convergence coefficient.** From numerical experiments, it is observed that the fraction of good sample points a_k converges exponentially to 1 during the consolidation phase, i.e., it approximately holds

$$1 - a_k \approx b \exp(-ck)$$

for all iteration steps $k \geq 0$. The coefficients b and c can be determined by a discrete least-squares fit for M iteration steps:

$$\sum_{k=0}^{M-1} |a_k - \widetilde{a}_k|^2 \to \min, \text{ where } \widetilde{a}_k = 1 - b \exp(-ck).$$

Figure 16 shows the fraction a_k of good sample points and the fitted curve $\widetilde{a}_k$ for Problem 3 for $d = 100$ dimensions and $N = 100$ samples. They are both in good agreement during the iteration. Since the cost per iteration is the number N of samples, we introduce the *convergence coefficient* c/N as a measure for the algorithm's convergence speed.

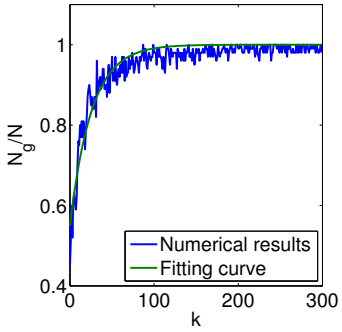
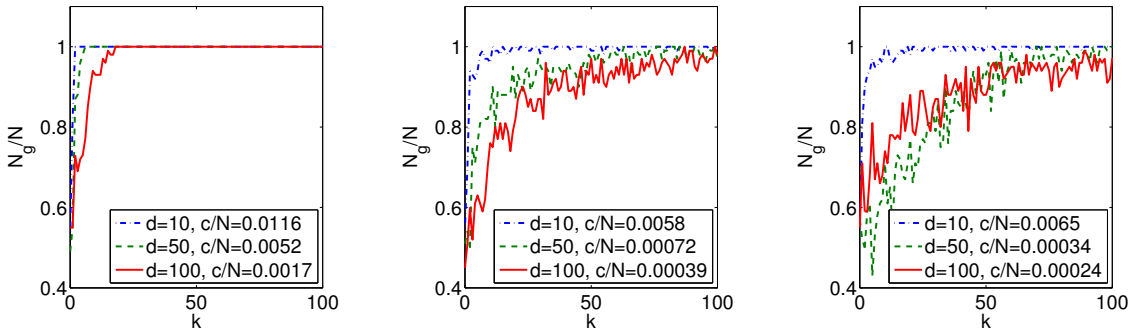


FIGURE 16. Convergence speed of the fraction of good sample points: Numerical results and fitting curve $\widetilde{a}_k$ for $d = 100$ and $N = 100$.

5.2. **Influence of the dimensionality.** The influence of the dimensionality on the convergence speed is illustrated in Figure 17 for Problems 2–4. Herein, the fraction of good sample points is plotted versus the number of iterations for different dimensions and $N = 100$ sampling points per iteration step. The convergence coefficient is found in the legend of the plots. For all problems, the convergence speed decreases when the dimension increases.



(a) (b) (c)

FIGURE 17. Fraction of good sample points versus the number of iterations for $N = 100$ for (a) Problem 2 (hyperbox), (b) Problem 3 (tilted hyperplane), and (c) Problem 4 (front crash).

The following mechanism describes the influence of dimensionality, which will be called *each bad sample point can be used for one dimension only*: If, for example, there are 100 sample points in a hyperbox and the fraction of good points is 80%, then 20 bad sample points are used to remove bad space. In 10 dimensions, there will be enough sample points to remove bad space in each dimension because we find two bad sample points for each dimension on average. The problem which arises in 100 dimensions is that the bad sample space can be removed in 20 dimensions at most because a bad sample point can be used to remove bad space in one dimension only. Bad space without bad sample points cannot be removed. Consequently, the optimum hyperbox will be overestimated in some dimensions. This is illustrated in Figure 18(a) for Problem 2 in $d = 2$ dimensions and one bad sample point. The grey area describes the bad space of the design space and the white area describes the good space. This fact explains that the algorithm converges slower in higher dimensions.

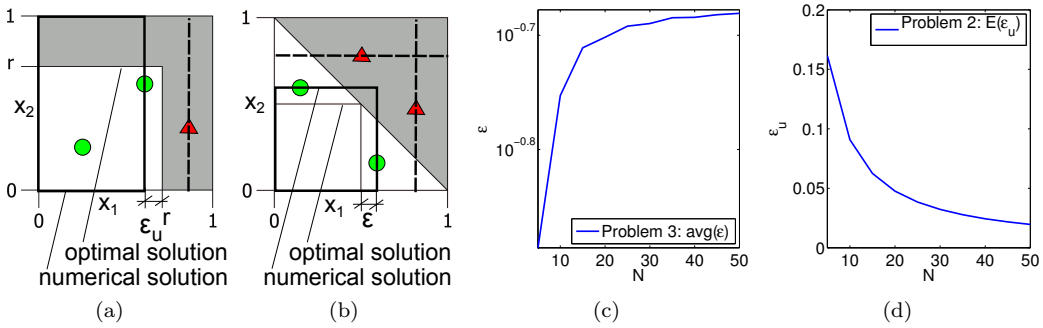


FIGURE 18. Mechanisms explaining over- and underestimation. (a) Problem 2. *Each bad sample point can be used for one dimension only* and *impossibility of boundary corrections*. (b) Problem 3. *Overestimation due to sparse sampling*. (c) Average avg(ε) of the measure ε for Problem 3. (d) Expectation of the underestimation measure ε_u for Problem 2.

5.3. **Influence of the number of sample points.** The influence of the number N of sample points per iteration on the convergence speed and the hyperbox volume is studied in this subsection for Problems 2–4. Every point in the diagrams in Figure 19 displays the mean of five calculations.

5.3.1. *Number of sample points versus convergence speed. Problem 2.* Figure 19(a) shows a maximum of the convergence coefficient for $d = 10$ and $d = 50$ dimensions. The reason for this maximum is that the fraction N_g/N of good sample points reaches at this point the value 1 in only one iteration step. For N larger than the peak location, $N_g/N = 1$ is also reached in only one iteration step, and c/N decreases. Left of the peak, the convergence coefficient increases as the number of sample points increases. This can be explained by the mechanism *each bad sample point can be used for one dimension only*, see Subsection 5.2. For each dimension, at least one bad sample point with $x_i \geq r$ has to exist to remove the bad space, see Figure 18(a). If the number of sample points increases, then the number of dimensions where bad volume can be removed increases. Consequently, the convergence speed increases until there are sufficiently many bad sample points, i.e., in every dimension at least one.

Problem 3. Figure 19(b) shows that the convergence coefficient increases as the number of sample points per iteration decreases, independently of the number of dimensions. If the number of sample points per iteration is small, then the volume which is removed per bad sample point is large, also the bad volume. To explain the mechanism, consider a two-dimensional sample for Problem 3 in the design space $[0, 1] \times [0, 1]$. Recall that the optimal solution is $x_1 = x_2 = 0.5$. Then, the distance between the optimal boundary location 0.5 and a sample point $\mathbf{x} = (x_1, x_2)$,

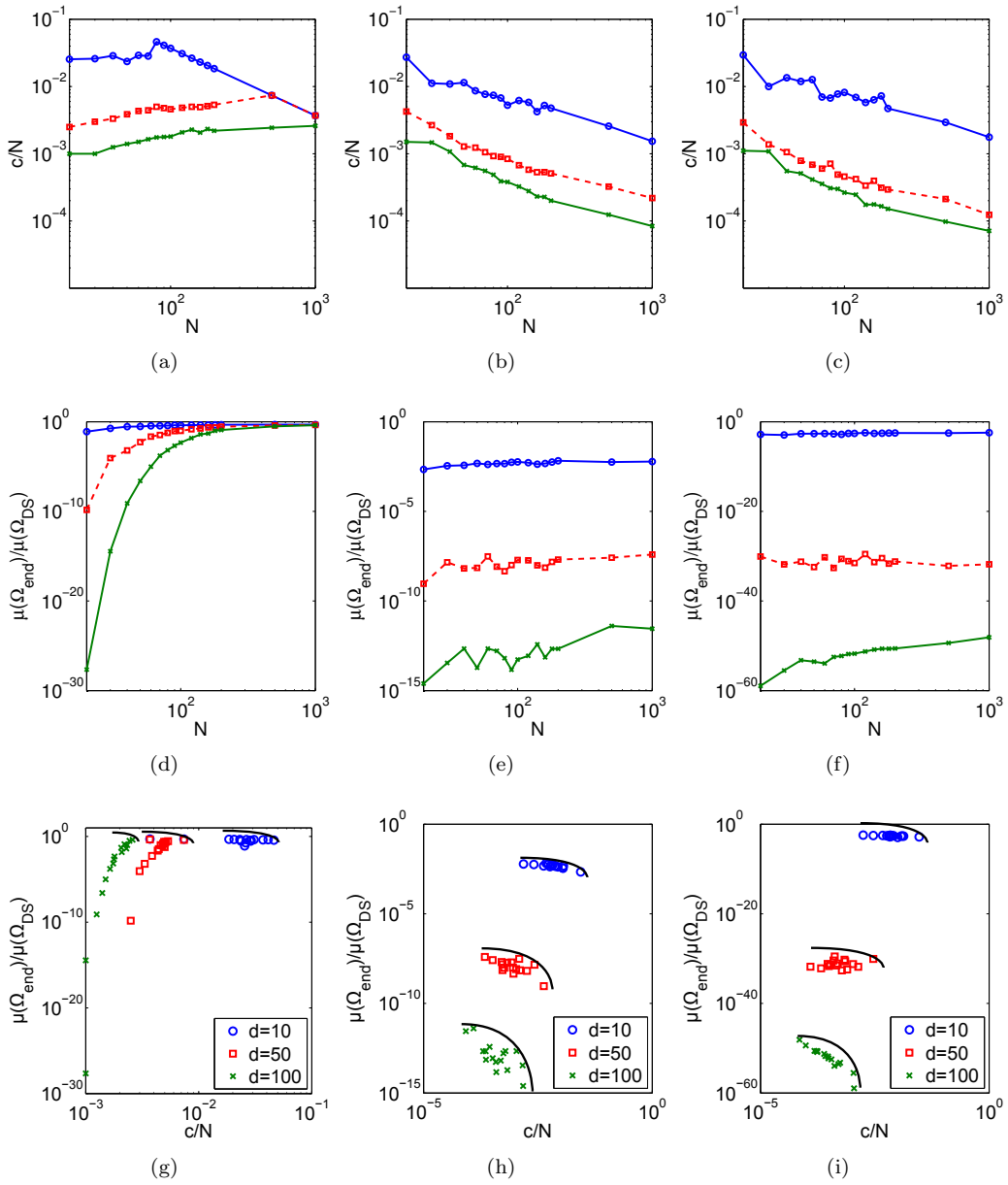


FIGURE 19. Convergence coefficient (first row), normalized volume (second row), Pareto frontier for the convergence coefficient and the normalized volume (third row) for Problem 2 (first column), Problem 3 (second column), and Problem 4 (third column).

where the boundary of the candidate hyperbox will be located, is calculated in dimension $i = 1$, i.e.,

$$\varepsilon = x_1 - 0.5.$$

In Figure 18(b), the measured distance ε is shown by four sample points. The distances for $L = 10\,000$ repetitions for each number N of sample points are averaged and plotted in Figure 18(c). The average of the measure ε is always larger than 0 and increases as the number N of sample points increases. Since $\varepsilon > 0$, the optimum hyperbox is overestimated, therefore, this

phenomenon is called *overestimation due to sparse sampling*. The larger the overestimation, the slower the convergence speed. Consequently, the convergence speed decreases as N increases.

Problem 4. Changing now to Problem 4 (front crash), according to Figure 19(c), a behavior can be observed that is similar to the one observed for Problem 3. The convergence coefficient c/N decreases when the number N of sample points increases. This indicates similar shapes of the boundaries separating good from bad space.

5.3.2. *Number of sample points versus volume.* In addition to the convergence speed of the fraction of good sample points, described by the convergence coefficient c/N, the volume $\mu(\Omega_{end})$ of the final hyperbox needs to be taken into account.

It can be observed in Figure 19(d) that for Problem 2 the normalized volume increases a lot as the number of sample points per iteration increases. Whereas, the corresponding diagram for Problem 3 (see Figure 19(e)) shows that here the volume increases only a bit when the number of sample points per iteration is increased. The same is observed for Problem 4, cf. Figure 19(f).

The increase of volume for larger N can be explained by *overestimation due to sparse sampling* which was explained in the previous Subsection 5.3.1, see also Figure 18(b). This effect shows that, with an increasing number of sample points N, the average of the considered distance ε increases, see Figure 18(c). If the number of sample points is small, the volume which is removed per bad sample point is large and the resulting volume is small. Therefore, the volume of the final hyperbox increases as the number of sample points increases.

Another reason for the increasing volume is the *impossibility of boundary corrections*: To explain this mechanism, Problem 2 is considered in one dimension. One sample with N sample points is made in the interval $[0, 1]$. Then, the expected distance between the optimal point $r = 0.5$ and the sample point x_j where the boundary will be after removing the bad sample points is

$$\mathbb{E}(\varepsilon_u) = \int_{[0,1]^N \setminus [0,0.5]^N} \min_j : \{0.5 - x_j : x_j < 0.5\} \, \mathrm{d}(x_1, x_2, \ldots, x_N) = \frac{1}{N+1} \left(1 - \frac{1}{2^N}\right).$$

Due to the rectangular boundaries of Problem 2, the optimum hyperbox will always be underestimated, see Figure 18(a), and ε_u is an underestimation measure. If the solution hyperbox boundary is larger than 0.5, then $\mathbb{E}(\varepsilon_u) = 0$. We observe in Figure 18(d) that the expectation $\mathbb{E}(\varepsilon_u)$ decreases as the number N of sample points increases. Therefore, the volume of the resulting hyperbox increases.

In Problem 2, the dependency of the final hyperbox volume on the number of sample points is greater than in Problem 3. Problem 2 has boundaries which are axis-parallel and, thus, the effect of the impossibility of boundary corrections is much stronger in Problem 2 than in Problem 3.

5.4. **Convergence speed versus hyperbox volume.** Typically, there is a conflict between the fast convergence and the volume size of the resulting hyperbox. For Problems 3 and 4, the convergence coefficient decreases whereas the resulting hyperbox volume increases upon increasing N. This conflict can be visualized by a Pareto frontier, see Figure 19. For Problem 2, this conflict exists only for large N, that is, right of the peak in Figure 19(a). For small N however, both, convergence coefficient and volume size, increase, see Section 5.3.

If a large volume is desired, as many sample points as possible have to be chosen. This will slow down the algorithm. If fast convergence is desired, as few sample points as possible have to be chosen. The smallest convergence coefficients are observed in Figure 19 for Problem 3 and 4 and for $d = 100$ dimensions as 10^{-4}–10^{-3}, corresponding to roughly ≈ 10000 function evaluations in Table 4.

The resulting hyperbox will be small. For computing the optimal number of sample points, the following procedure is proposed: Depending on the preference for speed or volume size, c/N is chosen using Figure 19. The total number of required evaluations for a particular choice of c/N is shown in Table 4 for an $a_0 = 50\%$, where a_0 is the fraction of the good space in the initial candidate hyperbox.

convergence coefficient	c/N	0.01	0.001	0.0001	0.00001
number of evaluations	NM	267	2661	26594	265928

TABLE 4. Convergence coefficient and the total number of function evaluations required for a fraction of the good space of 50% in the initial candidate hyperbox.

6. Conclusion

In the present paper, we analyzed an algorithm which identifies a hyperbox with maximum volume in the design space such that all designs inside this hyperbox are subcritical. The method can be applied to high-dimensional, nonlinear and noisy systems. For a design to be good, the choice of a parameter value within its assigned interval does not depend on the values of the other parameters as long as they are within their respective intervals. In this sense, the parameters are decoupled from each other. Robustness can be measured by the size of the resulting intervals. Moreover, intervals help to identify relevant parameters to improve a non-robust or critical solution. They may be combined with intervals of other disciplines – their cross sections are global solution spaces.

The algorithm determines a solution hyperbox provided that the initial guess contains enough good space. The algorithm's output is a numerical approximation of the exact solution and may therefore contain designs $\mathbf{x}$ that do not satisfy the imposed requirement $f(\mathbf{x}) \leq f_c$. However, the probability to be inaccurate in this sense can be computed and controlled by Theorem 3.3 independently of the number of dimensions. Final convergence is reached, when the probability for the resulting hyperbox to contain only good designs is sufficiently large. Nonetheless, for problems with disconnected regions, the solution hyperbox might only be locally optimal. It might extremely differ from the largest optimum hyperbox in the design space because the algorithm is only able to move the candidate hyperbox between disconnected regions if the growth rate is large enough.

Several benchmark problems were constructed to study the convergence of the proposed solution algorithm. The convergence in mean to the optimal solution was shown in low dimensions and for problems with rectangular boundaries in high dimensions. In Problem 3 (tilted hyperplane), the volume of the hyperbox found by the algorithm for high dimensions is very large compared to the optimum hyperbox. Nevertheless, the bad volume contained in the computed hyperboxes is small. If the widths of the intervals of a hyperbox are slightly larger than the widths of the intervals of the optimum hyperbox, then the volume of this hyperbox is considerably larger than the volume of the optimum hyperbox in high dimensions, that is, the available data become sparse. This effect reflects the curse of dimensionality. However, this observation does not affect the practical applicability since the fraction of the good space is still close to 100%. This was demonstrated by an engineering model for a front crash.

Furthermore, the convergence behavior of the consolidation phase of the algorithm was analyzed. The convergence coefficient was introduced to measure the convergence speed. It turns out that the convergence speed decreases as the number of dimensions increases. Finally, the conflict between the resulting volume and the convergence speed was studied for several high-dimensional optimization problems. The volume of the solution hyperbox increases as the number of sample points per iteration increases. However, the convergence coefficient decreases. This conflict can be visualized by a Pareto frontier. Mechanisms explaining this behavior were identified as *overestimation due to sparse sampling*, *impossibility of boundary corrections* and *each bad sample point can be used for one dimension only*. For Problem 3 (tilted hyperplane) and Problem 4 (front crash), the same convergence behavior was observed. This indicates that, in Problem 3, the boundary of the good space has a similar shape as in Problem 4.

7. Nomenclature

a	True fraction of good design space
a_k	Fraction of good sample points in iteration step k
c/N	Convergence coefficient
d	Number of dimensions
$f(\mathbf{x})$	Objective function
f_c	Critical threshold value
M	Total number of iteration steps
N	Number of sample points
N_g	Number of good sample points
I_i	Interval for variable x_i
x_i	Design variable i
$\mathbf{x}_i$	Vector of design variables x_i
x_i^{low}, x_i^{up}	Upper and lower limits of variable x_i
$\mathbf{x}^{low}$, $\mathbf{x}^{up}$	Vectors of upper and lower limits
ε	Error
μ	Size measure of box Ω
Ω	Hyperbox
Ω_{DS}	Design space

References

[1] N.M. Alexandrov, M.Y. Hussaini: *Multidisciplinary design optimization. State of the art*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[2] M. Allen, M. Raulli, K. Maute, D.M. Frangopol: *Reliability-based analysis and design optimization of electrostatically actuated MEMS*. Computers and Structures, 82:1007–1020 (2004).

[3] M. Beer, M. Liebscher: *Designing robust structures. A nonlinear simulation based approach*. Computers and Structures, 86(10):1102–1122 (2008).

[4] R.E. Bellman: *Adaptive control processes: a guide tour*. Princeton University Press, Princeton, 1961.

[5] E.A. Bender: *An introduction to mathematical modeling*. Dover Publications, New York, 2000.

[6] H.-G. Beyer, B. Sendhoff: *Robust optimization. A comprehensive survey*. Computer Methods in Applied Mechanics and Engineering, 196:3190–3218 (2007).

[7] J. Fender, F. Duddeck, M. Zimmermann: *On the calibration of simplified vehicle crash models*. Struct. Multidiscip. Optim., Springer, Berlin, Heidelberg, 2013.

[8] S. Gunawan, P.Y. Papalambros: *A Bayesian approach to reliability-based optimization with incomplete information*. Journal of Mechanical Design, 128(4):909–919 (2006).

[9] I. Hacking: *The emergence of probability*. Cambridge University Press, Cambridge, 1975.

[10] E.M.T. Hendrix, C.J. Mecking, T.H.B. Hendriks: *Finding robust solutions for product design problems*. European Journal of Operational Research, 92:28–36 (1996).

[11] M. Huang: *Vehicle crash mechanics*. CRC Press, Boca Raton, 2002.

[12] M. Kalsi, K. Hacker, K. Lewis: *A comprehensive robust design approach for decision trade-offs in complex systems design*. Journal of Mechanical Design, 123(1):1–10 (2001).

[13] H. Kerstan, W. Bartelheimer: *Innovative Prozesse und Methoden in der Funktionsauslegung – Auslegung für den Frontcrash*. Fahrzeugsicherheit, VDI-Berichte, volume 2078, VDI Verlag, Düsseldorf, 2009.

[14] J.J. Korte, R.P. Weston, T.A. Zang: *Multidisciplinary optimization methods for preliminary design*. AGARD CP-600, Future aerospace technology in the service of the Alliance, Paris, Vol. 3, pp. C40:1–10, 1997.

[15] H.W. Kuhn, A.W. Tucker: *Nonlinear programming*. Proceedings of 2nd Berkeley Symposium, University of California Press, Berkeley, California, pp. 481–492, 1951.

[16] M. Lehar, M. Zimmermann: *An inexpensive estimate of failure probability for high-dimensional systems with uncertainty*. Structural Safety, 36–37:32–38 (2012).

[17] D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, B.S. Lee: *An adaptive inverse multi-objective robust evolutionary design optimization*. Genetic Programming and Evolvable Machines, 7(4):383–404 (2007).

[18] J. Marczyk: *Stochastic multidisciplinary improvement: beyond optimization*. American Institute of Aeronautics and Astronautics, AIAA 2000–4929, 2000.

[19] Z.P. Mourelatos, J. Liang: *A Methodology for trading-off performance and robustness under uncertainty*. Journal of Mechanical Design, 128(4):1195–1204 (2006).

[20] W.L. Oberkampf: *Uncertainty quantification using evidence theory*. Stanford University, Stanford, 2005.

[21] S. Pannier, W. Graf: *Sensitivity measures for fuzzy numbers based on artificial neural networks*. Applications of Statistics and Probability in Civil Engineering, CRC Press, London, pp. 497–505, 2011.

[22] S. Pannier, W. Graf, M. Kaliske, K. Grossenbacher, M. Ganser, A. Lipp, M. Liebscher, H. Müllerschön: *Affecting reliability of deep drawing processes in early design stages*. Association for Structural and Multidisciplinary Optimization in the UK (ASMO-UK), 2006.
[23] G.-J. Park, T.-H. Lee, K.H. Lee, K.-H. Hwang: *Robust design. An overview*. AIAA Journal, 44(1):181–191 (2006).
[24] M.S. Phadke: *Quality engineering using robust design*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.
[25] C.M. Rocco, J.A. Moreno, N. Carrasquero: *Robust design using a hybrid-cellular-evolutionary and interval-arithmetic approach: a reliability application*. Reliability Engineering and System Safety, 79:149–159 (2003).
[26] A. Saltelli, K. Chan, E.M. Scott: *Sensitivity analysis*. Wiley, New York, 2000.
[27] K. Siebertz, D. van Bebber, T. Hochkirchen: *Statistische Versuchsplanung: Design of Experiments (DoE)*. Springer, Berlin-Heidelberg, 2010.
[28] I.M. Sobol': *Sensitivity estimates for nonlinear mathematical models*. Mathematical Modeling and Computational Experiment, 1(4):407–414, 1993.
[29] R. Stocki: *A method to improve reliability using optimal Latin hypercube sampling*. Polish Academy of Sciences, Poland, 2006.
[30] R. Storn, K. Price: *Differential evolution. A simple and efficient heuristic for global optimization over continuous spaces*. Journal of Global Optimization, 11:341–359 (1997).
[31] L.P. Swiler, A.A. Giunta: *Aleatory and epistemic uncertainty quantification for engineering applications*. Sandia Technical Report, SAND2007–2670C, 2007.
[32] G. Taguchi, E. Elsayed, T. Hsiang: *Quality engineering in production systems*. McGraw-Hill, New York, 1989.
[33] W. Witteman: *Adaptive frontal structure design to achieve optimal deceleration pulses*. Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles, United States, Washington DC, 2005.
[34] Y.G. Zhao, T. Ono: *A general procedure for first/second-order reliability method (FORM/SORM)*. Structural Safety, 21(2):95–112 (1999).
[35] M. Zimmermann, J.E. von Hoessle: *Computing solution spaces for robust design*. International Journal for Numerical Methods in Engineering, 94(3):290–307 (2013).

Lavinia Graff, BMW Group Research and Innovation Center, Knorrstr. 147, 80937 Munich, Germany
E-mail address: Lavinia.Graff@bmw.de

Helmut Harbrecht, University of Basel, Department of Mathematics and Computer Science, Spiegelgasse 1, 4051 Basel, Switzerland
E-mail address: Helmut.Harbrecht@unibas.ch

Markus Zimmermann, BMW Group Research and Innovation Center, Knorrstr. 147, 80937 Munich, Germany
E-mail address: markusz@alum.mit.edu