INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING

Int. J. Numer. Meth. Engng 2012; **00**:1–29

Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

Computing solution spaces for robust design

Markus Zimmermann[1]*, Johannes Edler von Hoessle[1]

[1]*BMW, Department of Vehicle Safety*

SUMMARY

In some engineering problems, tolerance to variation of design parameters is essential. In an early development phase of a distributed development process for example, the system performance should reach the design goal even under large variations of uncertain component properties. The tolerance to parameter variations may be measured by the size of a solution space on which the system is guaranteed to deliver the required performance. In order to decouple dimensions, the solution space is described as multi-dimensional box with permissible intervals for each design parameter. An algorithm is presented that computes solution spaces for arbitrary non-linear high-dimensional systems. Starting from a design point with required performance, a candidate box is iteratively evaluated and modified. The evaluation is done by Monte Carlo sampling and Bayesian statistics. The modification algorithm drives the evolution towards increasing box size. Robustness and reliability with respect to the required performance can be assessed without knowledge of the particular kind of uncertainty. Sensitivity to design parameters can be quantified by the widths of solution intervals. Designs failing to meet the performance requirement can be improved by adjusting parameter values to lie within the solution space. The approach is motivated and illustrated by automotive crash design problems. Copyright © 2012 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Robust design; optimization; structures; unertainty; solution space; high-dimensional systems

*Correspondence to: Markus Zimmermann, BMW, EG-3, 80788 Munich, Germany. E-mail: markusz@alum.mit.edu

1. INTRODUCTION

The crash performance of a vehicle depends on the behavior of several interacting components, such as structural members or restraint systems. In an early design phase, the design work is done on a *system level*, i.e., the performance of each crash-relevant component is specified such that the vehicle delivers the required system performance. Later, components are designed in detail according to their specifications on the *component level*. Detail parameters on the component level determine the component performances, which in turn determine the overall vehicle performance. A robust crash design ensures that the required system performance is reached even when the performances of some components differ from those of the initially intended design. It provides as much room as possible for

- optimization of components with respect to weight and cost,
- changes over the course of development in order to fulfill requirements from other disciplines, such as driving dynamics or acoustic behavior,
- limited manufacturing possibilities, and
- variations of the underlying detail parameters, such as the thickness of metal sheets, causing variations in the component performance.

The vehicle design is *uncertain* in the sense that the knowledge of its final state is limited. The earlier the design phase, the larger are the possible changes that may follow, i.e., the larger is the associated uncertainty.

Classical optimization techniques may be used to identify the vehicle design with best performance. Unfortunately, optimal designs are not necessarily robust and tend to be highly sensitive to parameter changes. Some authors believe that classical performance based optimality and robustness are inherently contradictory objectives, see [15].

Advanced optimization techniques, such as Robust Design Optimization and Reliability Based Design Optimization, seek optimal solutions that are less susceptible to failure under parameter variations, see [3] and [17]. Here, the variability of input parameters is prescribed. It is assumed

Prepared using nmeauth.cls DOI: 10.1002/nme

or measured, and then expressed by appropriate mathematical concepts, such as probability distributions, ranges of possible values or fuzzy quantities described by membership functions. Based on this, a robustness or reliability measure may be computed for every design point and included into the objective function. In exchange for worse performance, solutions become more robust or reliable, see [12], [9] and [19]. The result of analyses like this is the location of an optimum with respect to performance and robustness or reliability for a prescribed variability of input parameters. This is useful when the variability of parameters is known and can be described, e.g., as aleatoric uncertainty by a probability distribution. If the variability is not kown or can be modified and is therefore a degree of freedom, a different approach is necessary.

The method proposed here aims at finding a large *solution space* rather than one single optimal solution. The solution space is such that

- the vehicle delivers the required system performance, if all component performances lie in the solution space, and
- components are *decoupled*, i.e., each component performance can be assigned a permissible range that does not depend on the other components' behavior.

The solution spaces considered are multi-dimensional boxes (or hyper-cuboids) which can be expressed as product of intervals. The intervals are independent of each other and may be treated as component design goals, enabling distributed component development in separate departments.

The proposed method is fundamentally different from Robust Design Optimization and Reliability Based Design Optimization in that it

- seeks the intervals of a permissible design range rather than a single design and, therefore,
- uses interval boundaries as degrees of freedom rather than design parameters.

The concrete design that will eventually be realized within the design range is a result of the development process and not of numerical optimization.

Two categories of approaches that solve similar problems could be identified in the research literature. The first category is based on an analytical technique, called Interval Arithmetic, see [18].

Prepared using nmeauth.cls DOI: 10.1002/nme

The algorithm presented in [20], for example, applies Interval Arithmetic within an iterative optimization scheme to compute the feasibility of candidate boxes. Interval Arithmetic, however, limits the applicability of the algorithm, because the accuracy of the results depends on the problem and cannot be assessed for general cases. The second category of comparable approaches comprises Data Mining and Machine Learning techniques for extracting patterns and information from data, see e.g. [10], for the theory background see [22] and [8]. The use of these techniques for the purpose of this paper, however, depends on the degree to which the result can be expressed as a solution box, i.e., a space with decoupled parameters.

Support Vector Machines are well established in Machine Learning. In classification problems, they offer fast predictive functions for the evaluation of a design point to belonging to one of the two categories "meets the design goal" or "does not meet the design goal", see [5] or [8]. Unfortunately, these functions are in general not an appropriate tool to transform the original problem into an understandable structure, especially in high dimensions. In particular, they do not identify solution boxes in the input space. As an exception, solution boxes are constructed with Support Vector Machines in the context of rule extraction in [11], however, only for linearly separable data. The problem of identifying solution boxes for general problems remains unsolved.

Another learning technique is Cluster Analysis. It is applied in [2] to compute solution boxes based on one stochastic sample of design points. This approach is limited to problems with large solution spaces or extremely large sample sizes. If the resulting solution space is not significantly larger than the Nth fraction of the sampled space, where N is the number of sample designs, the data is too sparse to verify that it is in fact a solution space. For high-dimensional problems with many relevant input parameters, solution spaces tend to become extremely small, thus requiring a very large number of sample points[†].

The simple algorithm proposed in this article is an iterative sampling scheme that probes a box-shaped candidate (or hypothesis) region by stochastic sampling and then readjusts its boundaries in

[†]The solution box of the example problem in section 5 has a volume of less than 10^{-17} times the volume of the design space. The approach in [2] would require more than 10^{17} sample points.

Prepared using nmeauth.cls
DOI: 10.1002/nme

order to, first, remove designs with insufficient performance, and second, explore more input space that has not been probed before. According to [8], this procedure could be called a combination of query and on-line learning. An example of query learning is presented in [6], where Support Vector Machines are trained on a carefully selected data set in order to reduce computational cost. The approach presented here is similar, because sparse samples are used and refined iteratively by zooming into the good design space, rather than relying on one dense sample of the entire design space. Examples of on-line learning are described in [7] and [16], where the predictive quality of Support Vector Machines is increased by adding more data points step-by-step. The algorithm presented in this paper does online learning in the sense that it uses the information of new sample points in each iteration step so as to update the location of a candidate box. Despite the similarity with the established query and on-line learning techniques, however, the proposed algorithm is different in that it does not try to resolve the exact shape of the boundary of the good input space. Instead, it seeks one box-shaped solution space that does not include bad designs.

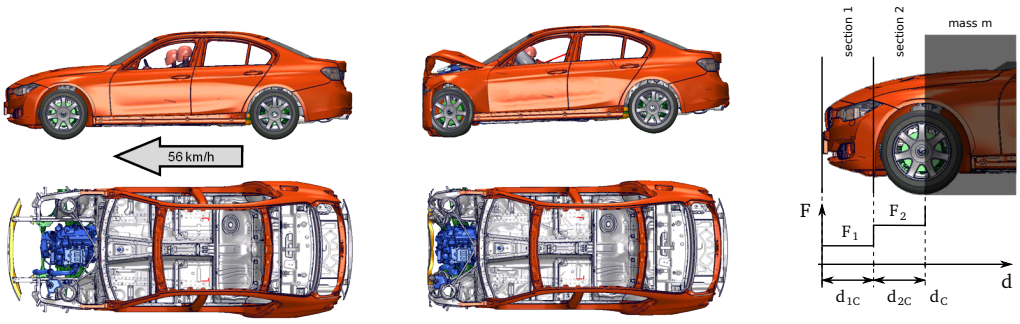
The paper is organized as follows: In the section 2, the search for box-shaped solution spaces is motivated by a simple example problem. A general problem statement is introduced in section 3. Section 4 explains in detail how the proposed algorithm works and how it is applied to the simple example problem. In section 5, a 36-dimensional problem is solved. Section 6 discusses the capabilities and the numerical performance of the algorithm.

2. A SIMPLE EXAMPLE PROBLEM

A design problem for a vehicle structure as shown in figure 1 is considered. The structure is to be designed for a front crash against a rigid barrier. The two sections represent structural components for each of which the deformation force is to be identified such that the vehicle delivers the required performance.

A simple model is used to describe the mechanical behavior of the vehicle structure. Elastic deformation is neglected and ideal plastic deformation is assumed. Deformation is possible at the deformation forces F_1 and F_2. There is no hardening, i.e., the deformation forces remain

Prepared using nmeauth.cls
DOI: 10.1002/nme



(a) (b)

Figure 1. (a) USNCAP Front Crash. (b) Model of front structure with force-deformation relationships for two sections

constant while deforming. The deformation measures read d_1 and d_2 for sections 1 and 2, the total deformation is $d = d_1 + d_2$. The deformations are limited to d_{1c}, d_{2c} and $d_c = d_{2c} + d_{1c}$, respectively. At the deformation limit, the force may become arbitrarily large in order to prevent further deformation. Only the front structure may deform: the rest of the vehicle is rigid. The vehicle mass m is concentrated on the rigid parts. The impact speed is v_0. The impact energy $\frac{1}{2}mv_0^2$ is assumed to be larger than the deformation energy of the first section $F_1 d_{1c}$, therefore, the second section will always deform.

The system design goals for the overall crash performance are, first, to keep the deceleration a of the passenger cell below a critical threshold level a_c (assuming that there is a solution, i.e., $a_c > \frac{v_0^2}{2d_c}$) and, second, to ensure an ordered deformation starting from the front of the vehicle. The design parameters are the two deformation forces F_1 and F_2.

The maximum deceleration is either $a = F_2/m$ when the impact energy is completely absorbed before $d = d_c$, i.e., $\frac{1}{2}mv_0^2 \leq F_1 d_{1c} + F_2 d_{2c}$, or is arbitrarily large otherwise. The required order of

Prepared using nmeauth.cls
DOI: 10.1002/nme

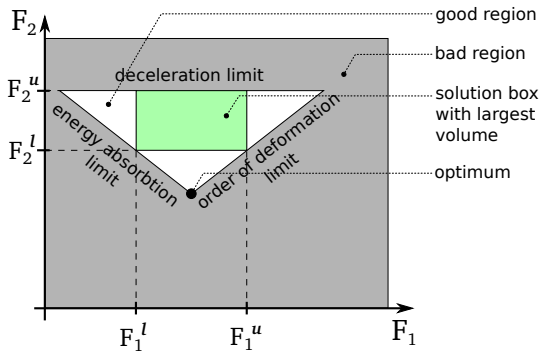


Figure 2. Solution space and the solution box with largest volume.

deformation may be realized when[‡] $F_1 \leq F_2$. The performance is measured by

$$z = f(F_1, F_2) = \begin{cases} 1 & \text{if } \frac{1}{2}mv_0^2 > F_1 d_{1c} + F_2 d_{2c} \\ 1 & \text{if } F_1 > F_2 \\ (F_2/m - a_c)/a_c & \text{otherwise.} \end{cases} \tag{1}$$

Good designs $(F_1,\ F_2)$ fulfilling both system design goals satisfy

$$f(F_1, F_2) \leq 0. \tag{2}$$

Figure 2 shows the complete solution space. The classical performance optimum that minimizes f lies at $(F_1, F_2) = (\frac{mv_0^2}{2d_c}, \frac{mv_0^2}{2d_c})$ with optimal performance value $z = (\frac{v_0^2}{2d_c} - a_c)/a_c$ and optimal deceleration $a = \frac{v_0^2}{2d_c}$. The best crash performance is achieved at a deceleration that remains constant over the entire course of the crash. This optimum is not robust, however. The immediate neighborhood of the optimum includes designs that violate condition (2) and, thus, fail to meet at least one system design goal. E.g., any increase of F_1 will cause the order of deformation to change, and any decrease of F_1 will prevent sufficient energy absorption.

[‡]When $F_1 = F_2$, the order of deformation is undefined. Assume in this case that an arbitrarily small imperfection in section 1 triggers the first deformation.

Prepared using nmeauth.cls
DOI: 10.1002/nme

For a component developer who tries to design one section with a required deformation force F_i, the optimum would not be a suitable component design goal, due to the lack of robustness. Using the entire solution space instead would also be impractical, as the tolerable range for F_1 would depend on the choice of F_2. A useful component design goal is of the form $F_i^l \leq F_i \leq F_i^u$, where i denotes the section number, and F_i^l and F_i^u denote the lower and upper boundaries of the design interval i, respectively. The boundaries are constant values, therefore, the component design goals are independent of the choice of deformation forces F_i and are decoupled in this sense.

Maximum flexibility for component design is provided if the intervals are as large as possible. The choice of intervals is not unique, and the size of the interval for F_1 depends on the choice of the interval for F_2. What interval is most useful to be maximized may depend on the problem considered. The approach proposed here is to maximze the product $(F_1^u - F_1^l)(F_2^u - F_2^l)$, and the resulting unique intervals are shown in figure 2.

The price to be paid for decoupling the deformation forces F_i is that some of the good design space cannot be used. The design goals require F_i to lie within their respective design intervals to ensure that the performance criterion (2) is satisfied. Therefore, the good region that is not included in the solution box in figure 2 is lost as accessible design space. If designs (F_1^*, F_2^*) from that region were admitted, e.g., with $F_2^* \leq F_2^l$, sufficient performance could not be guaranteed under the assumption that F_1^* may assume arbitrary values between F_1^l and F_1^u due to uncertainty. Note that the volume of the lost design space is minimized by maximizing the interval volume $(F_1^u - F_1^l)(F_2^u - F_2^l)$.

The example problem of this section is generalized in the following section.

3. GENERAL PROBLEM STATEMENT

3.1. Definitions

Design points or *designs* are represented by the vector $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, where p is the number of dimensions. The set of all possible design points defines the *design space* Ω_{ds}. The system's

Prepared using nmeauth.cls
DOI: 10.1002/nme

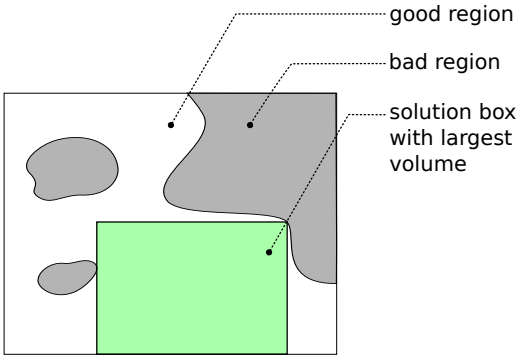


Figure 3. Solution box for a general problem. On the good region, design points satisfy $f(\mathbf{x}) \leq f_c$.

response at $\mathbf{x}$ is given by

$$z = f(\mathbf{x}) \tag{3}$$

with f being the *performance function*. The analytical form of f is in general not known, and z may have to be computed numerically. In typical optimization problems, the performance is optimized, i.e., a design $\mathbf{x}$ is sought such that z assumes an extreme value while other constraints are satisfied. In the approach presented here, the performance only has to be *sufficient* by satisfying the *performance criterion*

$$f(\mathbf{x}) \leq f_c \tag{4}$$

with the threshold value f_c. Constraints that would be treated separately in classical optimization can be included in this expression. Design points satisfying (4) are called *good*, those that do not are called *bad*. The design space can be divided into a good and a bad region, as shown in figure 3. The good region consisting of all good design points is the *complete solution space*. The shape of the complete solution space depends on the system's response and the performance criterion given by expressions (3) and (4), respectively. There are no restrictions on the function f, so the boundary separating good and bad design points may be arbitrary. Boundaries for the good range of one parameter x_i will in general depend on the values of other parameters x_j. In order to obtain interval boundaries for each parameter that are independent of other paramters only sub-spaces that

Prepared using nmeauth.cls
DOI: 10.1002/nme

are *boxes* are considered. A box is a product of intervals, i.e.,

$$\Omega = I_1 \times I_2 \times ... \times I_p, \tag{5}$$

with $I_i = [x_i^l, x_i^u]$ and the lower and upper boundaries x_i^l and x_i^u, respectively. The volume ratio of the good and the bad region enclosed in the box will be called *fraction of good designs*. A box including a specified fraction of good designs is called *solution box*. Figure 3 shows a solution box with good designs only. As there are infinitely many solution boxes for a problem, a size measure μ is used that quantifies the degree to which a box is a suitable solution to a particular design problem. One possible size measure is the volume

$$\mu(\Omega) = (x_1^u - x_1^l)(x_2^u - x_2^l)...(x_p^u - x_p^l), \tag{6}$$

however, other size measures are possible and may be more appropriate depending on the problem considered.

3.2. Problem statement for solution boxes with good designs only

Using the previously defined concepts, a simple problem statement that includes the relevant key ideas reads: For a given design space Ω_{ds}, a performance function f and performance threshold f_c, seek $\Omega \subseteq \Omega_{ds}$ such that

$$f(\mathbf{x}) \leq f_c \text{ for all } \mathbf{x} \subseteq \Omega \tag{7}$$

$$\mu(\Omega) \rightarrow \max. \tag{8}$$

This is an optimization problem, where μ is the objective function and expression (7) is a constraint. The degrees of freedom are the interval boundaries x_i^l and x_i^u, with $i = 1, 2, ..., p$.

Unfortunately, the evaluation of expression (7) is impossible for general problems on a continuous input space. If f is not known in closed-form and nothing is known about its structure, each design point included in a candidate box has to be evaluated in order to verify (7). The number of required function evaluations is infinite. If the input space is discrete, the number of necessary function evaluations is finite, but may be still large. E.g., for the problem in section 5, a complete scan of

Prepared using nmeauth.cls
DOI: 10.1002/nme

a box with 36 input dimensions and n discrete values for each variable would require n^{36} function evaluations.

3.3. Relaxed problem statement

The problem statement can be relaxed in order to make it applicable to general problems with an unknown function f. For practical design problems, it is typically *not* necessary to find a box including only good design points, in particular in an early development phase. It may be even desired to seek a box with a smaller fraction of good design points so as to obtain a large box. The final proof that a design meets all required design goals will always be done in a late development phase.

The Monte Carlo sampling technique (see [4]) can be used to estimate the fraction of good design points in a box as proposed in [14]. Sample designs are randomly and uniformly distributed over a candidate box and evaluated with respect to the performance criterion. If N is the total number of sample designs and m is the number of good sample designs, the ratio m/N is an estimate of the true fraction of good design points in the box. Ideally, candidate boxes are probed with many sample designs in order to obtain an accurate estimate. Each sample design, however, requires a function evaluation making large samples computationally expensive. In order to determine the required sample size, a statistical evaluation based on Bayesian inference can be applied. The accuracy of the estimate for a given sample size, and the probability that the true fraction of good design points lies within a specified range can be assessed as described in [14]. The procedure is briefly summarized here.

The fraction of good design points $\mathbf{x}$ in a box Ω is the ratio of the volume of good design space to the box volume, denoted by a. If a sample design is randomly chosen from the box, the probability for it to be a good design is a. If N sample designs are randomly chosen, the probability for m of them to be good is $P(m|a,N) = \binom{N}{m} a^m (1-a)^{N-m}$. Now assume that m and N are the result of random sampling as previously described, and nothing is known about a a priori. Then, the probability for

Prepared using nmeauth.cls
DOI: 10.1002/nme

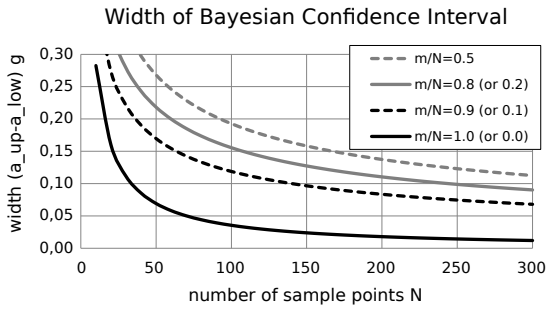


Figure 4. Width of the confidence interval of the estimated fraction of good designs for a confidence level of 95%. The confidence interval is smallest for observed m/N close to 1 or 0.

a to lie between a_l and a_u is

$$P(a_l < a < a_u \,|\, m, N) = \frac{\int_{a_l}^{a_u} t^m (1-t)^{N-m} \,\mathrm{d}t}{\int_0^1 s^m (1-s)^{N-m} \,\mathrm{d}s} = 1 - \alpha. \tag{9}$$

The interval $[a_l, a_u]$ is a *confidence interval* and $1-\alpha$ is the associated *confidence level*. For example, if all 100 sample designs are evaluated as good, then the true fraction of good designs of the box considered is larger than 97% at a 95% confidence level, because $P(97\% < a < 100\% \,|\, 100, 100) = 95\%$. This statement is independent of the number of dimensions p or the performance function f, in particular its convexity, smoothness or linearity. Widths of confidence intervals around the estimate m/N are shown for different sample sizes in figure 4.

The problem statement can now be rephrased as follows: For a given design space Ω_{ds}, a performance function f and performance threshold f_c, for given lower and upper bounds on the fraction of good designs a_l and a_u, and a critical confidence level $1-\alpha_c$, seek $\Omega \subseteq \Omega_{ds}$, such that

$$P(a_l < a < a_u \,|\, m, N) > 1 - \alpha_c \tag{10}$$

$$\mu(\Omega) \to \max, \tag{11}$$

where the number of good designs m is obtained by evaluating a Monte Carlo sample on Ω. This is again an optimization problem with objective function μ and constraint (10). The degrees of freedom are the interval boundaries x_i^l and x_i^u, with $i = 1, 2, ..., p$. The next section introduces an algorithm for this problem.

Prepared using nmeauth.cls
DOI: 10.1002/nme

4. A SEARCH ALGORITHM FOR SOLUTION SPACES

The proposed algorithm is an iterative optimization scheme that involves the evaluation and the modification of a candidate box. The evaluation is done by Monte Carlo sampling. The modification procedure removes bad regions and lets the candidate box evolve into good regions towards a box with larger size measure μ.

Underlying idea. An input space with unknown good and bad regions can be probed by sampling. The location of good and bad design points from the sample provide information that can be used to construct a candidate solution box, see figure 6. Good sample points *may* be included tentatively, however, bad sample points *must* be excluded. In modification step A, a group of candidate boxes is computed for a set of sample points from which the box with largest size measure μ is chosen. In modification step B, parameter intervals are extended so as to increase the size measure μ (the details will be provided later in this section). This procedure is independent of the number of parameters, therefore, it can be applied to arbitrary high-dimensional problems.

The resulting box is a new candidate box that has to be resampled, as some bad space may not have been detected in the previous sample. If much bad space was removed, the new candidate box will be smaller, providing a better resolution for the new sample. In this sense, the candidate box *zooms* into the good space. When the fraction of good sample points is sufficiently large according to (10), a solution box with the required estimate of good design space is obtained. The estimate is again not affected by the number of parameters, and the applicabilty to high-dimensional problems is not restricted.

Modification step A: Finding a box containing good sample designs only. Assume the Monte Carlo sample of a candidate box with N designs is known and m designs are good. If the fraction of good designs m/N is not large enough according to (10), a smaller box that contains only good sample designs can be constructed by moving the box boundaries such that the bad sample designs are

Prepared using nmeauth.cls
DOI: 10.1002/nme

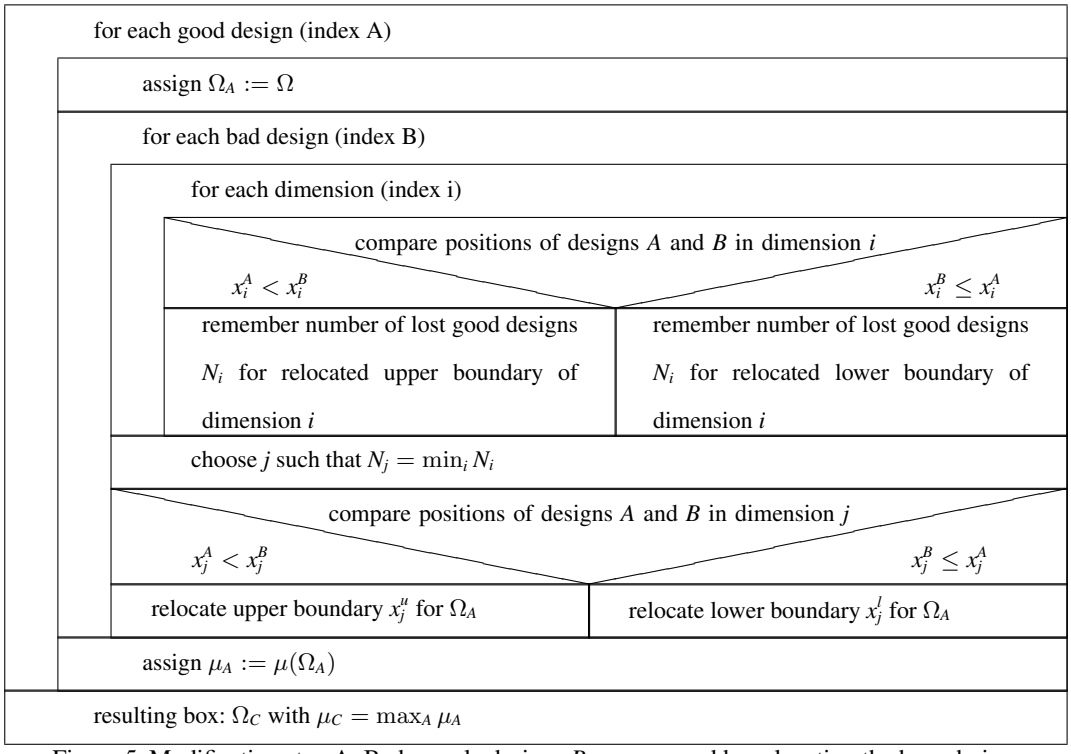


Figure 5. Modification step A. Bad sample designs B are removed by relocating the boundaries.

not included anymore. In this sense, bad designs are *removed* from the box. Removing bad designs should be done appropriately so as to obtain a box with a large measure μ.

This is done in three nested loops, as shown figure 5. The first one loops over all good designs (index A), the second one over all bad designs (index B) sorted by their performance value in descending order, and the third one over all dimensions (index i). For each good design A, a candidate box is constructed by removing all bad designs around it, thus, the box always contains design A. A bad design may be removed by moving one interval boundary of one dimension such that the bad design is not included anymore. The upper boundary is moved if $x_i^A < x_i^B$, the lower boundary otherwise. Which dimension is to be chosen, is evaluated in the third loop.

As the resulting box should be as large as possible, the loss of good regions is to be avoided. An estimate of the good region that is lost when a boundary is moved is the number of sample designs that will not be included in the box anymore. This number is counted for each dimension i, and the dimension with the smallest associated number lost is chosen. The new boundary will then be put on the coordinate value of the good design that is closest in that dimension to the bad design B. The

Prepared using nmeauth.cls
DOI: 10.1002/nme

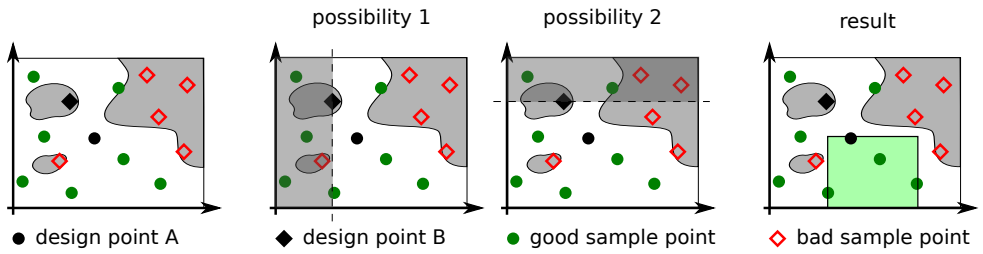


Figure 6. Relocation of interval boundaries in loop 2. There are two possibilites to remove sample point *B*. Possibility 2 is preferred over possibility 1, because less good designs are lost. Having all bad designs removed, the result of loop 2 contains only good designs.

procedure is illustrated in figure 6. The resulting box for design A is not necessarily the largest box possible, because the size of the good region that is removed can only be estimated. Also, the order by which bad designs are removed may have an effect on the result.

Having completed the three loops, there is a box for each good design A that contains no bad designs of the Monte Carlo sample anymore. The box with the largest value of μ is chosen as final result. Although all bad sample designs were removed from the box, it may nevertheless include bad regions of the input space that have not been sampled. In order to estimate the fraction of good designs in the new box, another Monte Carlo sampling is necessary.

Where the iteration starts. An intuitive approach to finding a solution box with a large size measure μ is to sample the entire design space Ω_{ds} and then apply modification step A. This approach is not practical for problems where the volume of solution boxes is very small compared to the volume of the entire design space, as is typically the case for high-dimensional problems. For the example problem in section 5, the volume ratio is 10^{-17}, as shown in figure 11. Sampling the design space would require a prohibitively large number of sample points in order to identify any solution box. Instead, the initial candidate box is chosen around a good design which is identified using classical optimization algorithms, such as differential evolution (see [21]).

Prepared using nmeauth.cls
DOI: 10.1002/nme

Modification step B: Enlarging the box. After modification step A has removed bad regions of the input space, modification step B probes all dimensions again in order to find larger solution spaces. Probing means that the boundaries of the candidate box are extended tentatively. The boundaries of dimensions that were not moved in modification step A due to bad designs are extended further to access more good input space. Those that were moved due to bad designs may not be limited by a bad region anymore, as boundaries of other dimensions were adjusted. Therefore, they are extended again. There are no assumptions regarding the structure of f, therefore after each modification step A, every dimension i is considered to be equally likely to provide more good input space. The basic algorithm proposed here extends every interval boundary of the candidate box by $g(x^u_{ds,i} - x^l_{ds,i})$, where g is the *growth rate*, and $x^u_{ds,i}$ and $x^l_{ds,i}$ are the upper and lower boundaries of the design space for dimension i, respectively. Bad regions that the candidate box has intercepted will be detected by Monte Carlo sampling (if they are sufficiently large) and then be removed in modification step A.

The growth rate is to be chosen such that the new candidate box created contains sufficiently many good sample designs. If the box is extended deep into a bad region and there is only a small number of good sample designs, the performance of modification step A will be decreased. At the same time, the growth rate should be sufficiently large to converge quickly to the optimal box. The growth rates used here are constant values which are chosen for each problem individually.

Complete algorithm: Phase I and II. The iteration process can be divided into two phases: In phase I, the solution box is to be enlarged as much as possible while keeping the fraction of bad input space at a reasonably low level. Therefore, modification steps A and B are applied. During phase II, the bad space is to be removed, thus, only modification step A is applied. The candidate box zooms into the good design space while increasing the fraction of good sample points to the desired level as given in expression (10).

Sampling in modification step A discovers bad regions only with a certain probability. In order to make sure that all bad regions are properly removed from the candidate box within one iteration step, one could either increase the sample size or run modification step A several times. In phase I,

Prepared using nmeauth.cls
DOI: 10.1002/nme

Classical optimization on Ω_{ds}. Identify one good design $\mathbf{x_0}$.	
The first candidate box Ω is constructed at $\mathbf{x_0}$ with zero volume.	
Phase I. While $\mu(\Omega)$ is changing:	
	Modification Step B: Extend candidate box.
	Compute Monte Carlo sample in Ω.
	Modification Step A: Remove bad sample designs.
Compute Monte Carlo sample in Ω.	
Phase II. While $m/N < a_c$:	
	Modification Step A. Remove bad sample designs.
	Compute Monte Carlo sample in Ω.

Figure 7. Complete Algorithm with classical optimization, phase I and phase II.

however, knowing the exact fraction of bad input space in the candidate box is not useful. Enlarging the candidate box in modification step B creates new bad regions to a degree that is unknown anyway. Therefore, after each modification step A, modification step B follows immediately. Running modification steps A and B in an alternating fashion makes the candidate box evolve towards a box with a larger measure μ, as shown in the examples of the following sections.

Once, μ does not change significantly anymore, the bad regions included in the candidate box are removed by running only modification step A, until expression (10) is satisfied. Figure 7 shows an overview over the complete algorithm.

Simple example problem. The algorithm is applied to the simple example problem of section 2. The size of the Monte Carlo samples is $N = 100$, the growth rate is $g = 0.2$. The first candidate box is constructed around the optimum, as shown in figure 8. Modification steps A and B are applied in phase I for 20 iteration steps. The volume increases rapidly, and the candidate box is extended into the good solution space. Then, the volume and the fraction of good design stagnate by oscillating at a constant level.

In 20 more iteration steps of phase II, all bad design points are removed. Only modification step A is applied, causing the volume to shrink and the fraction of good designs m/N to grow. The volume

Prepared using nmeauth.cls DOI: 10.1002/nme

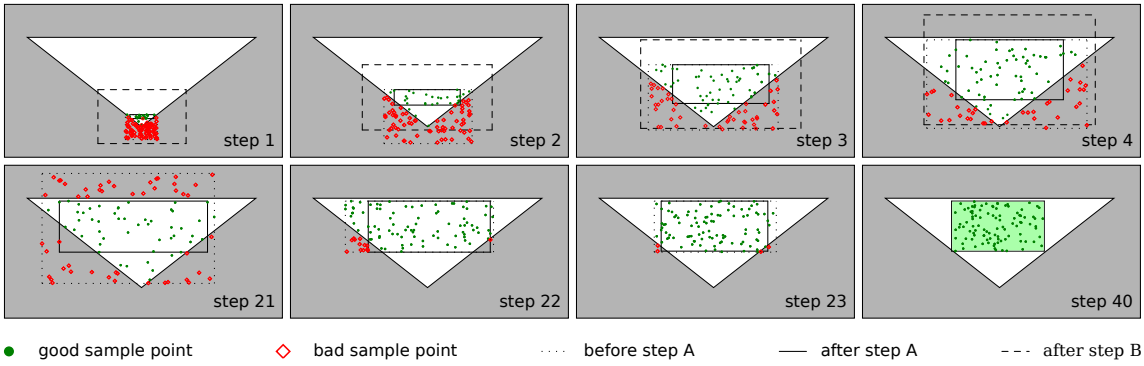


Figure 8. Evolution of the candidate box in phase I (steps 1-20) and phase II (steps 21-40). The result is shown at step 40.

	F_1^l	F_1^u	F_2^l	F_2^u
analytical result (kN)	291	516	516	628
numerical result (kN)	298	513	515	620
error (%)	-2.2	0.5	0.2	1.3

Table I. Analytical and numerical result of the simple example problem.

and the fraction of good designs finally converge to a constant value. The last box in figure 8 shows the result of the iteration.

Table I compares the analytical and numerical solution for $m = 2000\,\text{kg}$, $a_c = 32\,\text{g}$, $v_0 = 15.6\,\text{m/s}$ and $d_1 = d_2 = 0.3\,\text{m}$. The classical optimum that minimizes the performance function f lies outside the solution box at $(F_1, F_2) = (404, 404)\,\text{kN}$.

5. HIGH-DIMENSIONAL CRASH PROBLEM

5.1. Problem description

Another crash analysis problem is considered, based on a structural model as described in [13]. The model simulates a vehicle structure in a front crash as specified in the USNCAP test procedure

Prepared using nmeauth.cls
DOI: 10.1002/nme

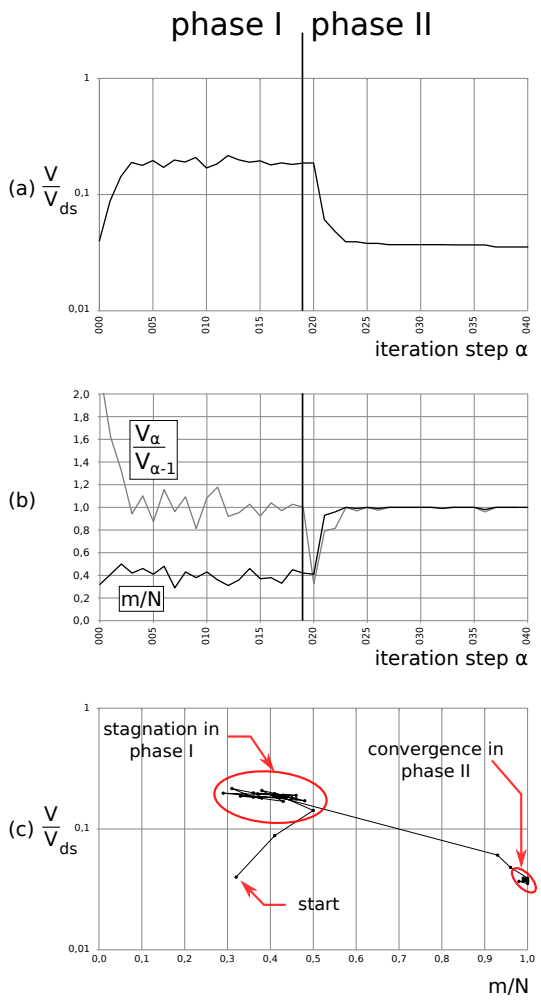
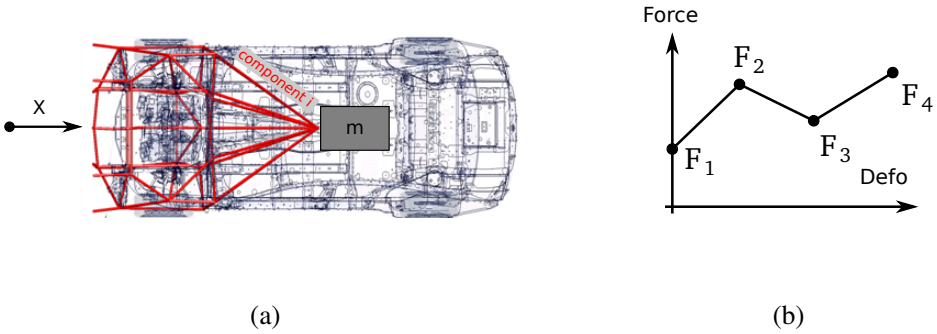


Figure 9. Simple example problem. (a) Normalized Volume, (b) fraction of good sample points m/N and volume ratio, and (c) normalized volume over m/N.

(see [1]), where the vehicle hits a rigid barrier wall at the speed of 56 km/h. The model is used to assess the structural performance and provides a mapping from the relevant component properties $\mathbf{x} = (F_i)$ to the system performance of the vehicle structure $z = f(\mathbf{x})$. System design goals are met when $f(\mathbf{x}) \leq 0$, as in section 2. For the development of the vehicle structure in the early development phase, a large solution space for the component properties F_i is sought. The solution space is to be specified as intervals. The required fraction of good designs in the solution space is at least 97%

Prepared using nmeauth.cls
DOI: 10.1002/nme



(a) (b)

Figure 10. (a) Structural Model for USNCAP front crash. (b) Force-deformation relationship for one component. The parameters F_i are compressive force levels at specified deformations.

with 95% confidence. This is established when the Monte Carlo sample of a candidate box has a fraction of good design points of $m/N = 100\%$.

5.2. *Model description*

The component properties F_i have a physical meaning that is derived from the relevant deformation mechanisms of the crash. The vehicle front structure is a cross-linked framework of longitudinal members (see figure 10). Positions where longitudinal members are connected to other members are called *nodes*. Members between two nodes are *structural components*. The deformation of one component is measured by the relative displacement of its two nodes in the x-direction (i.e., negative driving direction). Mass is distributed on the nodes, the mass of the passenger cell and the rear structure is attached to the center node. The performance of a structural component is measured by the force that is exerted at a particular deformation in the x-direction. As all deformations are assumed to increase monotonically over the course of the crash, no distinction between elastic and plastic deformation is necessary. The relationship between deformation u and force F for each component may be expressed by constitutive functions $F = \hat{F}(u)$ which include all relevant information that is necessary to compute the structural performance of the vehicle. The constitutive functions are approximated by discrete force values F_i at defined support points, as shown in figure

Prepared using nmeauth.cls
DOI: 10.1002/nme

10. The force values F_i of all components are the degrees of freedom that completely describe the design point $\mathbf{x} = (F_i)$.

The crash performance is measured by the deceleration of the vehicle and the order of deformation, as in the simple example problem, and, in addition, by the deformation of the passenger cell. It exceeds the required performance threshold when $f(\mathbf{x}) \leq 0$. The value of the performance function f is computed by an explicit time integration scheme, therefore no analytical form of f is available. It is known, however, that f is highly non-linear due to contact phenomena such as the engine block impacting the barrier. Therefore, any algorithm making use of linearity would be inappropriate.

The solution space for 9 structural components is computed, named A1, ..., C3 (see figure 12). Each component has a force-deformation relationship with 4 support points, and therefore 4 force values, that characterize its component performance. The design goal for each component is that its force-deformation relationship lie within the 4 associated intervals. The total number of design parameters, i.e. force values, that determine the system response is 36.

Component B3 is known to be difficult to design for a particular force-deformation relationship. Therefore, its intervals are fixed to be sufficiently large, i.e., they are not changed by modification step A. Other components need to compensate the enforced tolerance for component B3, and the resulting solution box will be smaller than one that is obtained without this constraint.

5.3. Results

The starting point of the iteration is the result of an optimization (differential evolution algorithm with 20 iterations and a population size of 100). It is not crucial that this design point is in fact optimal, it only has to satisfy the performance criterion. The growth rate for phase I is chosen to be 0.006, the Monte Carlo sample size is $N = 100$.

In the beginning of phase I, the volume of the candidate box increases rapidly by several orders of magnitude, as shown in figure 11. In the first 8 iteration steps, no bad designs occur in the candidate box, and the fraction of good designs remains at 100%. When the first bad designs are

Prepared using nmeauth.cls DOI: 10.1002/nme

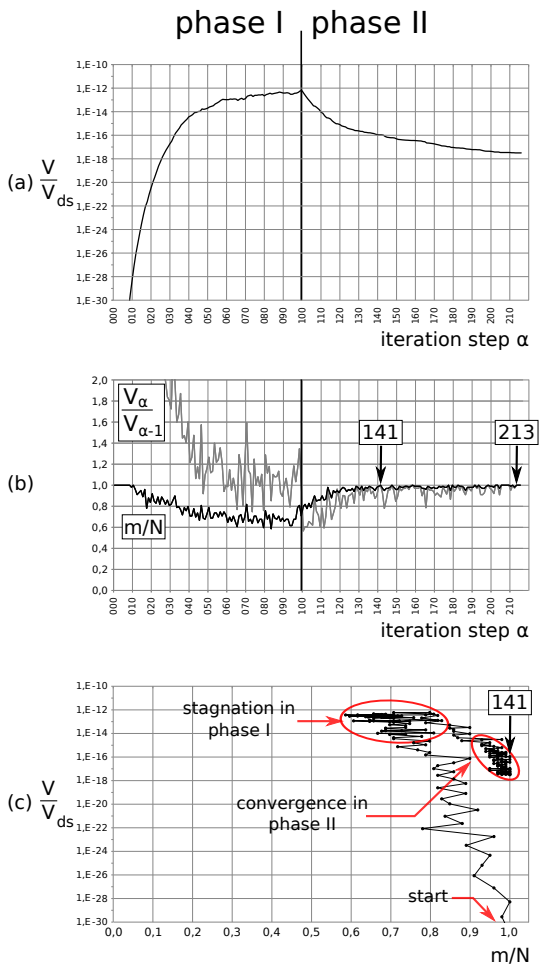


Figure 11. High-dimensional example problem. (a) Normalized Volume, (b) fraction of good sample points m/N and volume ratio, and (c) normalized volume over m/N.

encountered, the volume growth slows down and finally stagnates at approximately $10^{-12}V_{ds}$ (V_{ds} denotes the design space volume), and the fraction of good designs m/N oscillates at a level of 70%. In phase II, the volume of the candidate box is reduced, while the fraction of good designs increases. In iteration step 141, the first candidate box with $m/N = 100\%$ is obtained. For this box, the true fraction of good designs is at least 97% with 95% confidence and is therefore a solution that satisfies (10).

The iteration is continued in order to explore further convergence. In the subsequent iterations steps, bad designs are encountered and the volume is further reduced by more than an order of

Prepared using nmeauth.cls
DOI: 10.1002/nme

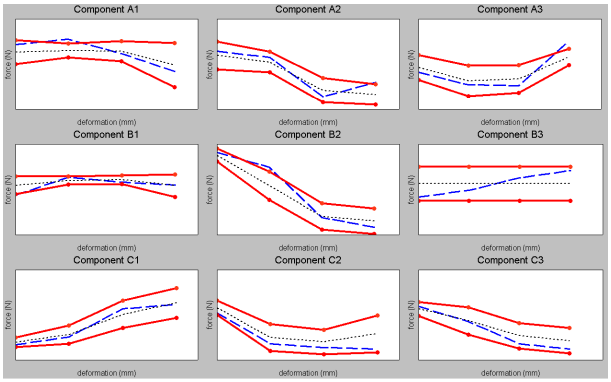


Figure 12. Solution box 141 shown as corridors for force-deformation-curves. The result of classical optimization (dashed lines) lies outside the corridors. The center lines are the robust target design (dotted lines).

magnitude. The fraction of good designs oscillates in the interval $[95\%, 100\%]$. This does not contradict the statistical statement about box 141 which is supposed include more than 97% good designs, because bad *and* good regions were removed. In iteration steps 213, 214 and 215, only good designs are encountered, i.e., the total number of good sample designs is 300. This implies that the true fraction of good designs is more than 99% with 95% confidence. In this sense, box 215 is better than box 141, however, it is smaller and it required significantly more computation time.

Figure 12 shows the solution box 141. For each component, there is a set of intervals serving as component design goal. Every component design goal is independent of the performance of other components. The result of the classical optimization, i.e., the starting point of the iteration, lies outside the box, therefore it is not the design to be chosen for best robustness. Instead, the target design for best robustness is the center line between the intervals. Critical components may be identified by their associated interval width. E.g., the first interval of component B2 is smaller than the intervals of component B3. Therefore, component B2 is critical for the overall performance and has to be designed more carefully than component B3.

In addition to robust design, the solution intervals may be used for sensitivity analysis of designs that fail to deliver the required performance. In crash analysis, it is often difficult to identify those components that cause system failure due to strong non-linear interaction. With the method proposed

Prepared using nmeauth.cls
DOI: 10.1002/nme

here, a component performance can be identified as insufficient, if its force-displacement curve lies outside the solution space intervals. The system performance can be improved by modifying the component accordingly.

5.4. Comparison of high- and low-dimensional problem

The 2- and 36-dimensional candidate boxes of the example problems from the previous sections, see figures 9 and 11, evolve in a similar way by exhibiting distinct periods of growth and stagnation in phase I, and a period of convergence in phase II. However, for 36 dimensions, more iterations are necessary than for 2 dimensions. In phase I, this can be explained by a smaller growth rate which is used to ensure that candidate boxes always include a certain fraction of good sample points. As the volume ratio of a candidate box before and after modfication step B may be as large as $(2g)^p$, growth rates have to be smaller for high-dimensional problems. In phase II, convergence is slower for high dimensions, because there are more boundaries to be relocated. Each bad sample point can be used only once for the relocation of an interval boundary, therefore, more sample points and thus more iterations are required.

6. DISCUSSION

The proposed algorithm always produces a result for any non-linear and high-dimensional problem, provided that one good design is known. This section discusses why it is applicable to arbitrary problems and how the results are limited with respect to validity and optimality. A comment on numerical performance is included in the end of this section.

Applicability to arbitrary non-linear and high-dimensional problems. No assumptions were made about the linearity or the dimensionality of problems to be treated with the proposed method. The underlying idea of the algorithm is independent of the number of parameters. Therefore, for the 2- and the 36-dimensional problem, the evolution processes of their candidate boxes are similar in that they exhibit a distinct growth, stagnation and convergence period, as explained in section 5.4. The validity of a solution box is also not restricted by the dimensionality of the problem, because

Prepared using nmeauth.cls
DOI: 10.1002/nme

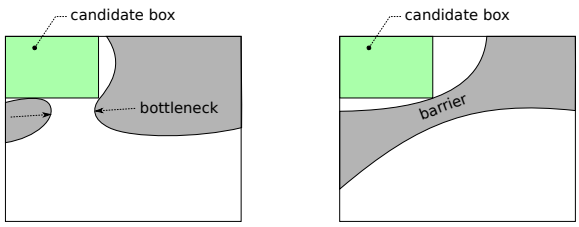
the estimate of the true fraction of good input space is independent of the number of parameters. For these reasons, the proposed method can be applied to arbitrary non-linear and high-dimensional problems.

Validity. The algorithm provides a solution space that is not guaranteed to include only good designs. The validity in the sense of expression (10) is established by Bayesian inference, providing a statistical estimate of the true fraction of good input space. The probability of an arbitrary design from the solution box to be good can be estimated by the fraction of good input space if no design is preferred to another, i.e., no a-priori information is available. If there are preferred designs, the probability of encountering bad designs in the input space will be unknown. It can nevertheless be made arbitrarily small by running more iterations of phase II.

Optimal shape. The result of the algorithm is a box-shaped solution space. This has the advantage that design intervals for one parameter are independent of the other parameter values, i.e., parameters can be decoupled. However, good design space is lost, as shown in figure 2 and explained in section 2. Design intervals could be extended in order to access more good design space, if the added bad designs were known to be sufficiently improbable to be realized. To assess this however, probability distributions for uncertain parameters would have to be known, which is typically not the case. Consequently, when using the solution box method, the price to be paid for flexibility in the design process is the loss of good design space. By maximizing the solution box volume, however, the volume of the lost space is minimized.

Global optimality. The resulting solution box is not necessarily globally optimal in the sense of (11). The iteration starts at a candidate box that is constructed around a good design point. In phase I, the neighborhood is probed in order to extend the boundaries of the candidate box. Bad regions in the design space could be a barrier or a bottle neck that would block the evolution of a candidate box towards the globally optimal solution space, see figure 13. In this sense, the algorithm searches for a local optimum. Finding a globally optimal solution box requires a complete scan of the design space, e.g., by uniform sampling, if the properties of the functions f are unknown. This can only be successful, if the size of a Monte Carlo sample is sufficiently large, in particular larger

Prepared using nmeauth.cls
DOI: 10.1002/nme



(a) (b)

Figure 13. Bad region preventing the candidate box from entering a larger good region by forming a bottleneck (a) or a barrier (b).

than $V_{ds}/V(\Omega_{opt})$, where V_{ds} and $V(\Omega_{opt})$ denote the volume of the design space and the optimal solution box, respectively. In crash problems that are similar to the example problem in section 5, affordable[§] Monte Carlo samples of the design space often do not include a single good design point. Therefore, the globally optimal solution box remains unknown, and, consequently, $V(\Omega_{opt})$ is too small to be discovered by Monte Carlo sampling. Whether or not the result of the algorithm is globally optimal cannot be assessed.

Local optimality. Modification step A produces a candidate box that may not have the best size measure μ. This is because Monte Carlo sampling was used to estimate the location of good and bad regions. In addition, bad sample points are removed one by one in only one order (in order of their performance value). The order, however, has an effect on the result. As in phase II only modification step A is applied, candidate boxes can only shrink, and a sub-optimal boundary modification cannot be corrected. A necessary condition for local optimality is that modification step A always produces a box with best measure μ, which could be accomplished by a different algorithm.

Numerical performance. For each iteration step, computation time is consumed, first, by creating data using stochastic sampling and, second, by updating the candidate box in the modification steps A and B. The numerical complexity of creating a Monte Carlo Sample and computing f for each

[§]Affordable means here $\sim 10^4$ function evaluations.

Prepared using nmeauth.cls DOI: 10.1002/nme

sample point is $O(N)$. Modification step A is $O(N^2p)$. Modification step B can be neglected, as it is $O(p)$ and uses only fast operations.

The numerical performance of modification step A and Support Vector Machines are similar under particular circumstances. For example, modification step A and the Bunch-Kaufmann training algorithm have the same complexity, when most support vectors are at the upper bound and the number of support vectors is close to the number of training points, see [5, page 147].

The total computation time of the proposed algorithm depends linearly on the number of iteration steps ν, which will be affected by the shape of the good space and by the dimensionality of the problem. The 2-dimensional example problem required on the order of 10 iteration steps, see section 4, whereas for the 36-dimensional problem from section 5, on the order of 100 iteration steps were necessary. The total computation time will be proportional to νN, if the evaluation of f is expensive and creating the data dominates the computational performance. If f is inexpensive and modification step A dominates, the total computation time will be proportional to $\nu N^2 p$.

7. CONCLUSIONS

Rather than optimizing one particular design for robustness or reliabilty, the approach presented in this paper provides an answer to the question: What is a large region in the input space where designs meet the required performance goal? Maximizing this region improves robustness and flexibility for component design that is naturally subject to uncertainty. In order to decouple parameters, regions that can be described as boxes are sought. Each permissible interval for one parameter is independent of other parameter values and can therefore serve as design goal for component development in a distributed development process during which the final design is chosen. The good design space outside a solution box is lost. This is the price that is paid to decouple parameters.

Designs with optimal performance are typically not included in the solution box, i.e., robustness and performance optimality are contradicting objectives. For a bad design, the intervals of the solution box provide concrete information about sensitivity. They identify, first, what parameter has

Prepared using nmeauth.cls
DOI: 10.1002/nme

to be changed, and, second, how it has to be changed in order to make a design reach the required performance.

An algorithm is proposed to compute solution boxes for any non-linear high-dimensional system. Based on Monte Carlo sampling and Bayesian inference, the fraction of good designs in a candidate box is evaluated. In an iterative scheme, the algorithm extends boundaries tentatively in every dimension to make it evolve towards a larger box. Then, it removes bad design regions to increase the fraction of good designs. Global and local convergence to the optimal solution box are not ensured. The validity of the solution box can be controlled by the number of iterations steps and the sample size. The true fraction of good designs can be assessed with arbitrary accuracy by a confidence interval at a defined confidence level for arbitrary dimensionality.

REFERENCES

1. National Highway Traffic Safety Administration. *Laboratory Test Procedure TP208-13*. Office of Vehicle Safety Compliance, NVS-220, 400 Seventh Street, Washington, DC 20590, 2005.
2. M. Beer and M. Liebscher. Designing robust structures - a nonlinear simulation based approach. *Computers and Structures*, 86, 2008.
3. H. Beyer and B. Sendhoff. Robust optimization - a comprehensive survey. *Comput Methods Appl Mech Engrg*, 196, 2007.
4. K. Binder. Monte-carlo methods. In G. L. Trigg, editor, *Mathematical Tools for Physicists*. Wiley-VCH, 2006.
5. C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 1998.
6. C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proc. 17th Int. Conf. Machine Learning (ICML2000)*, 2000.
7. G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Fourteenth conference on Advances in Neural Infomation Processing Systems, NIPS*, 2001.
8. N. Cristiani and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.
9. I. Doltsinis and Z. Kang. Robust design of structures using optimization methods. *Comput Methods Appl Mech Engrg*, 19, 2004.
10. U. Fayyad, G. Platetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3), 1996.

Prepared using nmeauth.cls
DOI: 10.1002/nme

11. G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *Proceedings of The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, 2005.
12. Y. Jin and B. Sendhoff. Trade-off between performance and robustness: an evolutionary multiobjective approach, in: C. fonseca. In *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO*, 2003.
13. H. Kerstan and W. Bartelheimer. Innovative prozesse und methoden in der funktionsauslegung – auslegung fuer den frontcrash. In *Fahrzeugsicherheit, VDI-Berichte*, volume 2078. VDI Verlag, Duesseldorf, 2009.
14. M. Lehar and M. Zimmermann. An inexpensive estimate of failure probability for high-dimensional systems with uncertainty. *Struct Saf*, 2011.
15. J. Marczyk. Stochastic multidisciplinary improvment: Beyond optimization. *AIAA*, 4929, 2000.
16. M. Martin. On-line support vector machines for function approximation. Technical report LSI-02-11-R, Software Department, Universitat Politecnica de Catalunya, Spain, 2002.
17. B. Moeller and M. Beer. Engineering computation under uncertainty - capabilities of non-traditional models. *Computers and Structures*, 86, 2008.
18. R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliff, NJ, 1966.
19. M. Mourelatos and J. Liang. A methodology for trading-off performance and robustness under uncertainty. *J Mech Des*, 4, 2006.
20. C. Rocco and J. Moreno. Robust design using a hybrid-cellular-evolutionary and interval-arithmetic approach. *Rel Eng System Safety*, 19, 2003.
21. R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 1997.
22. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 2 edition, 1999.

DOI: 10.1002/nme