Identification of Crashworthy Designs Combining Active Learning and the Solution Space Methodology

Paolo Ascia, Stefano Marelli, Bruno Sudret, Fabian Duddeck

ETH zürich

Identification of crashworthy designs combining active learning and the solution space methodology

P. Ascia, S. Marelli, B. Sudret and F. Duddeck

Chair of Risk, Safety and Uncertainty Quantification
Stefano-Franscini-Platz 5
CH-8093 Zürich

Risk, Safety & Uncertainty Quantification

Data Sheet

Journal:	ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering
Report Ref.:	RSUQ-2024-010A
Arxiv Ref.:	–
DOI:	10.1115/1.4066621
Date submitted:	July 15, 2024
Date accepted:	October 23, 2024

Identification of crashworthy designs combining active learning and the solution space methodology

Paolo Ascia [*1], Stefano Marelli[†2], Bruno Sudret[‡2], and Fabian Duddeck [§1]

[1] *TUM School of Engineering and Design, TUM, Germany*
[2] *Chair of Risk, Safety and Uncertainty Quantification, ETH Zürich, Switzerland*

October 23, 2024

Abstract

This study introduces a novel methodology for vehicle development under crashworthiness constraints. We propose coupling the Solution Space Method (SSM) with Active Learning Reliability (ALR) to map global requirements, i.e., safety requirements on the whole vehicle, to the design parameters associated with a component. To this purpose, we use a classifier to distinguish between the design that fulfills the requirements, the *safe domain*, and those that do not, the *failure domain*. This classifier is trained on finite element simulations, exploiting the learning strategies used by ALR to efficiently and precisely identify the border between the two domains and the information provided on these domains by the SSM. We then provide an exemplary application where the efficiency of the method is shown: the safe domain is identified with 270 samples and an average total error of 2.5 %.
The methodology we propose here is an efficient method to identify safe designs at a comparatively low computational budget. To the best of our knowledge, there is currently no methodology available that can identify regions in the design space that result in designs satisfying the local requirements set by the SSM due to the complexity and strong non-linearity of crashworthiness simulations. The proposed coupling exploits the information of SSM and the capabilities of ALR to provide a fast mapping between the global requirements and the design parameters, which can, in turn, be made available to the designers to inexpensively evaluate the crashworthiness of new shapes and component features.

1 Introduction

Designing a new vehicle is a challenging process. It requires the expert knowledge of designers and engineers to find a balance between contrasting requirements, such as the need for lightweight

[*]paolo.ascia@tum.de
[†]marelli@ibk.baug.ethz.ch
[‡]sudret@ethz.ch
[§]duddeck@tum.de

vehicles and the safety of the passengers. To help the developers navigate these contrasts, a popular strategy is to use the tools developed in the field of systems engineering. These tools generally help to better formulate the requirements. They, however, do not help identify possible solutions, i.e., a physical design of the product, to fulfill all requirements. In this work, we consider how the challenge of identifying possible requirements-worthy designs can be solved when using the Solution Space Method (SSM) (Zimmermann and De Weck, 2021) as the support tool from the field of systems engineering.

To handle the overall complexity of vehicle design, SSM cascades the requirements from a global level to the local level of vehicle components. With requirement on a global level, or *global requirement*, we refer to a condition the whole vehicle must fulfill. Whereas, by a *local requirement*, we mean the requirement each single vehicle component must fulfill so that the final assembly meets the overall global requirement. For example, in the field of passive safety, the global requirements to mitigate the risks of injury of the passengers linked to a high-speed crash are stated in the protocols released by safety rating agencies, like the National Highway Traffic Safety Administration (National Highway Traffic Safety Administration, b). The local requirements for passive safety are, for example, how much energy each component must dissipate and how much each component is allowed to deform. If all local requirements are met by each component, then the global requirements are met, and the passengers will not suffer major injuries in a high-speed crash. In SSM, the requirements are cascaded down to a set of components deemed relevant for the specific passive safety study. Therefore, SSM reformulates the risk mitigation strategies of the whole vehicle on a local level. Thanks to SSM, each component can be developed to fulfill the global requirements, thus being *crashworthy*, independently of the development of the other components. This is achieved by defining how much each component must deform, d_i in Fig. 1, and the force range each component must oppose during the impact, the white areas in Fig. 1. Therefore, the local requirements are presented in the form of force ranges over a defined deformation length, called *corridors*. An example of a typical set of corridors for a whole vehicle is later presented in Section 5.1.

By cascading global requirements, the SSM delivers to the designers and engineers a condition on the performance of the components under development (Zimmermann and Von Hoessle, 2013; Zimmermann et al., 2017). However, it does not provide any information about the structure or geometry of the design of the components. Hence, the SSM is limited by the unknown and generally complex mapping between local requirements and actual designs. This gap, to our knowledge, is nowadays filled by the expertise of the designers themselves, but no research has proposed quantitative ways to estimate such a mapping. With the methodology we present in this paper, we fill this gap by introducing a quantitative method to identify the regions of the design space that map into the safety corridors defined by SSM. This is known in machine learning as the pre-image problem (Kwok and Tsang, 2004). By identifying the pre-image of the corridors, we cascade the global requirements once more on the physical design of components instead of stopping on the performance level of the components. This step identifies specific structural

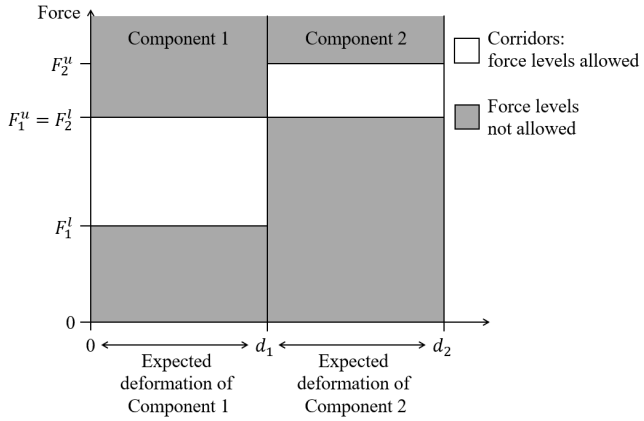


Figure 1: Example of corridors computed by the solution space method for crashworthiness. d_1 and d_2 are the expected deformation lengths for each component, while $F_{1,2}^l$ and $F_{1,2}^u$ are the lower and upper bounds of the computed force ranges, i.e., corridors. The corridors, meaning the crashworthy force ranges, are depicted by the white areas.

properties of a component, hereinafter called *design parameters*, that render the performance of the component itself fulfill the corridors of SSM. By doing so, we provide a tool to the engineers and developers to help them explore the potential of their new concepts and designs.

We provide a classifier that can be directly used to identify which set of design parameters yields a solution that fulfills the local requirements of the corridors and which does not, following the approach first introduced by Abbiati et al. (2022). We refer to the first type of sets as *feasible configuration* or *crashworthy* and to the others as *infeasible configuration*. By identifying the feasible design, we help developers and engineers design crashworthy components. This information is useful to, for example, constrain an optimization problem for crashworthiness and better exploit the design parameters available to the developers. To identify the crashworthy designs efficiently, we adopt and adapt, for the first time for crashworthiness to our knowledge, some of the core concepts from the field of active learning reliability (ALR) (Abbiati et al., 2022; Teixeira et al., 2021; Moustapha et al., 2022a). In classical ALR, a surrogate-model-based classifier is adaptively constructed from an expensive computational model, to efficiently identify small areas of the parameter space that result in the failure of an engineering system. We use a similar strategy to identify regions of the parameter space that result in feasible configurations. We first re-formulate the local requirements defined by SSM as a limit-state function, and then use the ALR algorithm to learn and identify the feasible areas in the domain of the design parameters. Instead of calculating the probability of failure of the system, we rather use the ALR algorithm to efficiently train a classifier that can identify at no added computational cost the crashworthiness of an unseen design. This classifier can then be used to constrain an optimization problem to design crashworthy components or to help developers unlock the full potential of their conceptual designs in relation to crashworthiness. This approach is related to the state-of-the-art in surrogate-aided reliability-based design optimization (RBDO, (Moustapha and Sudret, 2019)),

with the main difference being that the safety constraints are provided by SSM. By doing so, SSM allows an increase in the overall design flexibility compared to solving a secondary nested reliability problem (Zimmermann and Von Hoessle, 2013).

The main contributions in this work are:

- Reformulating the corridors of SSM as an equivalent reliability problem;
- Using state-of-the-art active learning methods to map corridor spaces back into the design space;
- Provide a fast classifier trained on the underlying physics-based model that can be used during the design phase to quickly assess design compliance with the SSM corridors.

The paper is organized as follows: Section 2 provides an overview of the relevant literature on the SSM. Section 3 introduces the basic concepts of reliability and the active learning strategies used in the field. Section 4 presents the new method, and Section 5 showcases an application on a realistic case study related to the re-design of the frontal crash-box of the 2013 Honda Accord (National Highway Traffic Safety Administration, b). Lastly, in Section 6, we comment on the results and discuss the methodology.

2 The Solution Space Method

The SSM belongs to the family of systems engineering tools used to support the development process. The method cascades a set of global requirements to a new product on a sub-system or even a set of single components as a set of local requirements. The purpose of this cascade is not only to simplify the development of a new product but also to allow the development of each component or sub-system independently of the others. Although the method is now applied in many different fields, like robotics (Krischer and Zimmermann, 2021) or driving dynamics (Vogt et al., 2019), we focus on the field of crashworthiness.

In the field of crashworthiness, the SSM was first introduced by Zimmermann and Von Hoessle (2013). More precisely, it was introduced to help fulfill the crashworthiness requirements imposed by the passive safety protocols. From its introduction, two different approaches have then been adopted in the crashworthiness field: the direct (Fender, 2014; Fender et al., 2017) and the indirect one (Graff, 2013). The first is completely based on an analytical approach to describe mathematically the product, its components and expected behavior, and to cascade the global requirements. The second, instead, is based on sampling a model of the product and cascading the global requirements, i.e., defining the local requirements for each component, by testing them on the samples of the said model. We focus on the direct approach for crashworthiness. We invite the reader to consider the work of Graff et al. (2016); Graf et al. (2018) for a more detailed explanation of the indirect approach.

According to the definition given by Fender et al. (2017), a *solution space* is the space defined as

$$\Omega = \left[x_i^l, x_i^u\right] \subset \mathbb{R}^n, \tag{1}$$

where x_i^l, and x_i^u belong to the n-dimensional vector $\mathbf{x}$ of performance parameters whose mapping to the expected behavior ($\mathbf{x} \mapsto y$) fulfills all global requirements. The vector $\mathbf{x}$ describes a set of properties of each component making up the final product.

This solution space is then optimized to find the biggest space in which all x_i are orthogonal to each other (Daub et al., 2020). In other words, the biggest possible hypercube is placed inside Ω to define a subset where the properties of different components are independent of each other. According to Daub et al. (2020), we define a subspace of the solution space

$$\Omega_c = \left[x_1^l, x_1^u\right] \times ... \times \left[x_n^l, x_n^u\right] \subset \Omega, \tag{2}$$

where the Cartesian product is imposed on the ranges of each x_i. The resulting ranges are called *corridors*.

In the direct method, the mapping of the performance parameters to the expected behavior is generally done by a simplified analytical model (Lange et al., 2019), as we will see later for the crashworthiness case study. Thanks to the simplified modeling and the definition of corridors, SSM has become an industry-standard tool to support development. On the one hand, the simplified modeling allows the computation of the corridors to be very efficient and computationally affordable. On the other hand, the definition of corridors allows for more flexibility during development (Zimmermann and Von Hoessle, 2013). Both the convenience and flexibility of the method are desirable characteristics that brought the industry to adopt the method (Zimmermann et al., 2017). We, therefore, choose to adopt this methodology so that we can evaluate the local crashworthiness requirements quickly and economically while allowing us to define a full range of possible crashworthy designs, i.e., areas of feasibility and infeasibility.

For crashworthiness, $\mathbf{x}$ is a vector of force levels $[F_1, ..., F_n]$ the components are expected to oppose during a crash. Therefore, the solution space Ω is defined in the space described by the force level ranges $\left[F_i^l, F_i^u\right]$. Consequently, also the corridors are an expression of the ranges on these force levels: $\Omega_c = \left[F_1^l, F_1^u\right] \times ... \times \left[F_n^l, F_n^u\right]$. Imposing the Cartesian product means that, as long as the forces opposed by the components are inside the range $\left[F_i^l, F_i^u\right]$, the crashworthiness of the vehicle is always guaranteed and each component can be designed independently.

3 Reliability analysis and active learning

Reliability analysis is a field of uncertainty quantification devoted to assessing the safety and reliability of engineering systems. It provides a collection of advanced methods for the estimation of the probability of failure of a system, due to the inherent uncertainty in its manufacturing, operating or environmental conditions. At the core of reliability analysis lies the concept of *limit*

state function, an abstract function implicitly defined through its sign: it takes positive values when the system is operating nominally, and negative otherwise.

More rigorously, consider an $M-$dimensional vector $\boldsymbol{x} \in \mathbb{R}^M$ with joint probability distribution $\boldsymbol{x} \sim f_{\boldsymbol{X}}$, that represents the uncertainty of environmental and system parameters under consideration. Consider also a computational model $y = \mathcal{M}(\boldsymbol{x})$ that calculates numerically a response quantity of interest (QoI) of the system for the set of parameters $\boldsymbol{x}$, e.g. a finite-element model (FEM) evaluating the maximum force or displacement in a component during a collision. A valid limit state function is any function $g(\boldsymbol{x})$ that takes negative values when $\boldsymbol{x}$ results in a system operating outside its intended conditions. Arguably, the most used class of limit state functions is that related to the exceedance of critical material parameters:

$$g(\boldsymbol{x}) = \rho_{\text{adm}} - \mathcal{M}(\boldsymbol{x}), \tag{3}$$

where ρ_{adm} is a the maximum admissible value of the QoI Y before the system fails. A limit state function partitions the input domain $\Omega_{\boldsymbol{X}}$ into two classes: a *safe domain* $\mathcal{D}_S$, that includes all the subsets of $\Omega_{\boldsymbol{X}}$ such that $g(\boldsymbol{x}) > 0,\ \forall \boldsymbol{x} \in \mathcal{D}_S$, and a *failure domain* $\mathcal{D}_F$ that satisfies instead $g(\boldsymbol{x}) \leq 0,\ \forall \boldsymbol{x} \in \mathcal{D}_F$.

Within this framework, the probability of failure P_f of the system is simply defined as the probability that the limit state function takes negative values, given the joint distribution $f_{\boldsymbol{X}}$:

$$P_f = \mathcal{P}(g(\boldsymbol{x}) \leq 0) = \int_{\mathcal{D}_F} f_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{4}$$

The estimation of the integral in Eqn. (4) is, in general, rather complex, as the failure domain $\mathcal{D}_F$ is only known implicitly through the sign of the limit-state function, which in general requires costly evaluations of the computational model $\mathcal{M}$. Several approaches are available to estimate it in the literature. Some of these are based on the approximation of the limit-state function with linear functions or with other semi-analytical methods (Ditlevsen and Madsen, 1996; Lemaire et al., 2009). The accuracy of these methods highly depends on the number of samples considered in the approximation. Therefore, applying these methods is usually computationally intensive because they generally require a high number of evaluations.

3.1 Active learning strategies for efficient sampling

To overcome the generally high computational effort required by the classical methods for estimating $\mathcal{D}_F$, the last decade has seen the steady growth of a class of methods that capitalizes on recent advances in machine learning and surrogate modeling, namely active-learning-based reliability analysis (Teixeira et al., 2021; Moustapha et al., 2022a).

Active learning in reliability analysis refers to the process of evaluating the computationally intensive limit-state function on a relatively small set of points, determined iteratively, that maximize the estimation accuracy of P_f at each iteration (Teixeira et al., 2021; Moustapha et al., 2022a).

The principle behind this class of methods, which belongs to the more general family of *greedy algorithms*, is to iteratively construct an inexpensive-to-evaluate approximation (a.k.a. a surrogate model) of the limit state function $\widehat{g}(\boldsymbol{x}) \approx g(\boldsymbol{x})$, based on a comparatively small number of full model evaluations, known as the *experimental design*. The choice of samples to add to the experimental design is based on maximizing a so-called *learning function* $\mathcal{L}(\boldsymbol{x})$, which associates to each point in the input domain a score related to the expected improvement in the accuracy of the P_f estimator. A vast body of literature is dedicated to the specific choice of learning functions for reliability analysis, and the user is referred to both Teixeira et al. (2021) and Moustapha et al. (2022a) for a more in-depth discussion. Because the failure probability P_f in Eqn. (4) only depends on $\text{sign}(g(\boldsymbol{x}))$, active-learning-based approximations tend to focus the expensive computational model evaluations only in the areas close to the safe-to-failure domain transition $g(\boldsymbol{x}) = 0$, commonly known as the *limit state surface*. In other words, central to all active learning methods is an accurate classifier of the safe and failure domains, trained at the lowest possible cost.

Despite the obvious differences between the plethora of methodologies available in the structural reliability literature, all active learning-based reliability methods share the same overall algorithmic structure (Moustapha et al., 2022a):

1. Train an initial surrogate model of the limit-state function $\widehat{g}(\boldsymbol{x})$ based on a small sample $\mathcal{X}_{\text{ini}} = \left\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N_{\text{ini}})}\right\}$ that follows the joint input distribution $f_{\boldsymbol{X}}$, with $N_{\text{ini}} \sim \mathcal{O}(10^1)$, known as the *initial experimental design*.

2. At each iteration i, Estimate an approximate probability of failure $P_f^{(i)}$ based on the current surrogate, as well as its associated confidence bounds and accuracy measures, if available. This estimation is typically performed using simulation-based methods, due to the extremely low costs of evaluating the surrogate over a large set of input parameters.

3. Minimize a suitable *learning function* (LF) over the input domain to identify the next point (or set of points) to be added to the current experimental design. A learning function is a generic mapping $j : \mathbb{R}^M \to \mathbb{R}$ that associates to each point in the input domain a value of information that is suitable for the learning goal at hand. Such value of information is typically related to the expected changes in misclassification probability of the surrogate limit state function $\widehat{g}(\boldsymbol{x})$, or to the reduction of the variance in the estimation of $P_f^{(i+1)}$.

4. Add the chosen point $\boldsymbol{x}^* = \min_{\boldsymbol{x} \in \Omega_{\boldsymbol{X}}} LF(\boldsymbol{x})$ to the current experimental design, and train a new surrogate of the limit state function.

5. Repeat steps 2-4 until some stopping criterion is met, i.e., convergence is achieved, or the available computational budget is exhausted.

Once the algorithm has completed, its final estimates of P_f and its confidence bounds are returned.

A large body of literature is available on the many different configurations of each of these steps, but an in-discussion on their performance is outside the scope of this paper and has been extensively covered by Moustapha et al. (2022a).

3.2 Active learning reliability for training classifiers

While not extensively used in reliability analysis applications (if not for failure mode identification, see e.g. the work of Sauder et al. (2019)), a byproduct of all ALR algorithms is the final trained surrogate model $\widehat{g}(\boldsymbol{x})$, which provides a powerful classifier for the identification of $\mathcal{D}_F$ and $\mathcal{D}_S$, without the need to run additional expensive computational model evaluations. Outside the field of reliability analysis, the use of ALR algorithms as tools to build a standalone classifier has been recently introduced by Abbiati et al. (2022), where the algorithm in Section 3 has been used to guide an experimental campaign to characterize the buckling behavior of steel plates.

A key difference between the use of ALR within reliability analysis vs. pure classifier training lies in the joint distribution of the input parameters $f_{\boldsymbol{X}}$, which represents the actual parametric and operational uncertainty in the former, while it only defines the allowed parameter space in the latter. In particular, all the parameters of the classifier are considered independent and uniformly distributed between the minimum and maximum bounds defined by the application under investigation.

In the crashworthiness field, methods like SSM already provide a set of criteria on each component — *local requirements* — to understand whether or not a design fulfills the safety requirements on the whole vehicle — *global requirements*. However, the local requirements returned by the SSM are in the form of a set of ranges of force levels that a component is required to oppose during deformation. Consequently, to understand if the design of component k fulfills the ranges $\left[F^l_{i,k}, F^u_{i,k}\right]$, it must be tested. Testing, generally, is rather expensive since it is performed with computationally intensive finite element simulations.

In contrast, building a standalone classifier using the ALR method is a cost-effective solution to finding designs that fulfill the corridors of the SSM. As shown by Abbiati et al. (2022), a relatively small number of evaluations is required to identify the failure areas in a given domain. Therefore, we can train a similar classifier to identify which designs yield a set of force levels inside the given corridors $\left[F^l_{i,k}, F^u_{i,k}\right]$ and which ones do not. Therefore, training such a classifier can significantly reduce the costs of developing new components for crashworthiness. In this paper, we propose, for the first time to our knowledge, a methodology to build this type of classifier that provides a tool to identify crashworthy designs of any given component. From a more practical point of view, we exploit the local requirements defined by the SSM to define the limit state function used by ALR to learn which combinations of design parameters are crashworthy and which are not.

4 Solution-space-based crashworthiness classifier

Our goal is to train a classifier capable of identifying crashworthy design inside the design domain using ALR and the SSM. We use SSM to define the local crashworthiness requirements of a component and then use ALR to train a classifier capable of distinguishing between designs that fulfill these local requirements and those that do not. The use of ALR for this application, though, requires some small changes. Normally, the algorithm is used to estimate the probability of failure of a given engineering system. Instead, we use the active learning capabilities of this algorithm to train a surrogate that accurately reproduces the sign of the limit state function, which directly translates to the two states of the classifier.

Firstly, we need to define a suitable function g that encodes the relation between the corridors and the force levels measured in a crash test of a component. Notice that, when formulating $g(\boldsymbol{x})$, the vector $\boldsymbol{x}$ contains a set of design parameters of a component that are controlled during the test, such as the shell thickness and the impactor mass. These are not to be confused with the performance parameters of the solution space, i.e. the force levels $F_1, ..., F_n$.

According to the definition of the corridors of the SSM, see Section 2, given a component k and set of force levels $\mathbf{F}_{\text{msr}} = [F_{1,k,\text{msr}}, ..., F_{n,k,\text{msr}}]$ measured during testing, a design is deemed crashworthy when every measured force $F_{i,k,\text{msr}}$ satisfies:

$$F^l_{i,k} \leq F_{i,k,\text{msr}} \leq F^u_{i,k}. \tag{5}$$

We can split Equation (5) into two separate equations. The first,

$$F_{i,k,\text{msr}} - F^l_{i,k} \leq 0, \tag{6}$$

defines a condition in which the term $F_{i,k,\text{msr}} - F^l_{i,k}$ is negative when the corridor is violated — the measured force is lower than the allowed level — and, vice versa, it is positive when the corridor is fulfilled.

The second equation relates instead to the upper limit. In the equation

$$F^u_{i,k} - F_{i,k,\text{msr}} \leq 0, \tag{7}$$

the term $F^u_{i,k} - F_{i,k,\text{msr}}$ is negative when the measured force is above the allowed level, i.e., violating the corridors of the SSM. Otherwise, $F^u_{i,k} - F_{i,k,\text{msr}}$ is positive. If the condition in Eqn. (5) is satisfied, then, both $F_{i,k,\text{msr}} - F^l_{i,k}$ and $F^u_{i,k} - F_{i,k,\text{msr}}$ are positive. If instead one of the conditions in Equations (6) and (7), then $F_{i,k,\text{msr}} - F^l_{i,k}$ or $F^u_{i,k} - F_{i,k,\text{msr}}$ is negative. Therefore, to define a function $g(\boldsymbol{x})$ according to the definition in Section 3, we consider the minimum value between the terms in Equations (6) and (7):

$$h(\boldsymbol{x})_{i,k} = \min\left(F_{i,k,\text{msr}} - F^l_{i,k}, F^u_{i,k} - F_{i,k,\text{msr}}\right), \tag{8}$$

for each i-th measured force level of component k.

On top of the conditions listed so far, we must define an extra function $h_{n+1,k}(\boldsymbol{x})$ that relates to the expected deformation used in Eqn. (10) and defined in the step-wise simplification introduced by Lange et al. (2019). According to them, a component cannot deform more than a pre-defined length. Therefore, the deformation length d_{msr} measured during the crash test, cannot exceed a pre-defined value d_{max}. Therefore, since $d_{\text{msr}} \leq d_{\text{max}}$, we can define an extra condition, the $i+1$-th one, for defining the limit state function,

$$h_{n+1,k}(\boldsymbol{x}) = d_{\text{max}} - d_{\text{msr}}. \tag{9}$$

Finally, to define a limit state function $g(\boldsymbol{x})$ that accounts for all $(n+1)$-th conditions of component k, we consider the minimum among all $h_{n+1}(\boldsymbol{x})$. This way, the value of $g(\boldsymbol{x})$ is negative if at least one of the i-th force ranges is violated or if $d_{\text{msr}} \geq d_{\text{max}}$, corresponding to the classical definition of a series system (failure occurs if any of the constraints is violated). Instead, if all i-th force ranges are fulfilled and the deformation length is smaller than d_{max}, then $g(\boldsymbol{x}) \geq 0$.

We feed the values of $g(\boldsymbol{x})$ to the ALR algorithm to learn $\mathcal{D}_S$ and $\mathcal{D}_F$ in the domain of design parameters. In other words, the algorithm follows the steps described in Section 3.1 to learn an approximation $\widehat{g}(\boldsymbol{x})$ of the limit state function $g(\boldsymbol{x})$. In the end, the learned function is used as our classifier. Given a set of design parameters $\boldsymbol{x}$, if $\widehat{g}(\boldsymbol{x}) \geq 0$, the design fulfills all corridors, thus the crashworthiness requirements. Otherwise, if $\widehat{g}(\boldsymbol{x}) \leq 0$, at least one of the corridors is violated; thus, the crashworthiness requirements are not met. The overall process is visualized in the flowchart of Fig. 2.

5 Application to a realistic design test case

To help showcase the potential of the methodology we are proposing, we present an application example in this section. To this purpose, we consider as a starting point the example presented by Daub et al. (2020) as a practical application of the SSM. The authors imagine re-designing the 2013 Honda Accord, available from the National Highway Traffic Safety Administration (National Highway Traffic Safety Administration, a). We, then, consider the frontal crashbox of the Honda Accord considered by Daub et al. (2020) and apply our methodology.

5.1 Application of the SSM to the 2013 Honda Accord

The application of the computation of the corridors for the 2013 Honda Accord study presented by Daub (2020) studies how the vehicle is expected to behave in a front impact with a rigid barrier and full overlap, according to the protocols of the National Highway Traffic Safety Administration (National Highway Traffic Safety Administration, b).

We first need to identify the sub-system responsible for the crashworthiness of the vehicle during a frontal impact. In the Honda Accord, seven components are expected to dissipate most of the

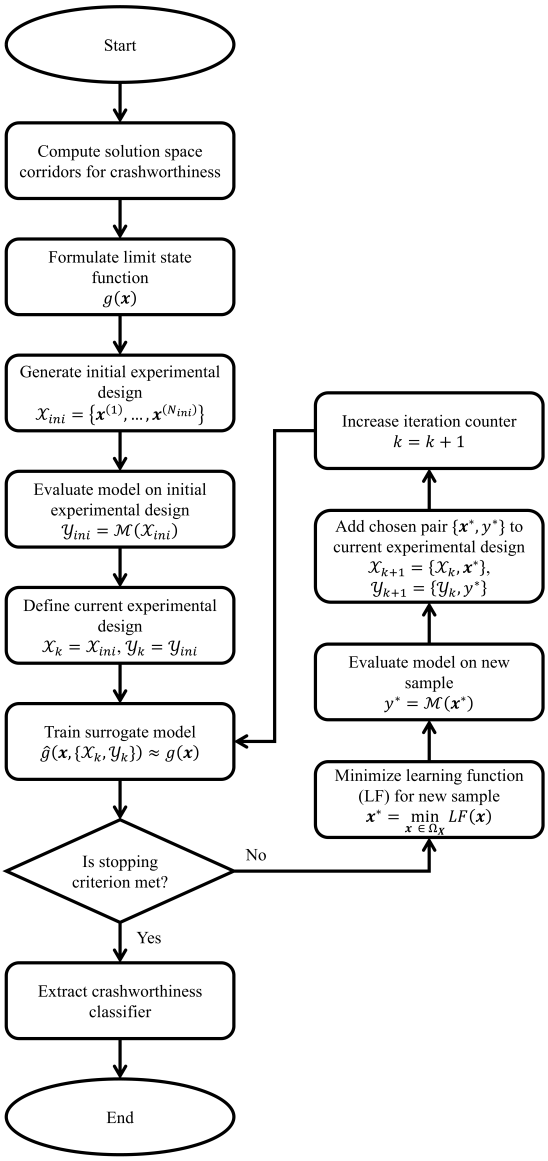


Figure 2: Flowchart with the overview of the entire process for training the crashworthiness classifier.

energy of the car during the impact. These components are divided into two load paths that deform at the same time. Then, to formulate the model itself, the properties of the components are considered for the stepwise simplifications necessary to represent the car and the crash performance (Lange et al., 2019). These simplifications correlate the properties of the vehicle, like mass and velocity at the beginning of the impact, with the total deformation of each component. This correlation then defines the so-called deformation space model shown in the intermediate step of Fig. 3. In the deformation space model, the deformation length of each component is also divided into several sections. For example, with reference to Fig. 3, Component 1 is divided into nine sections, while Component 4 in only two. Each section is then associated with one of the force levels composing the vector of performance parameters $\mathbf{F}$. The combination of force

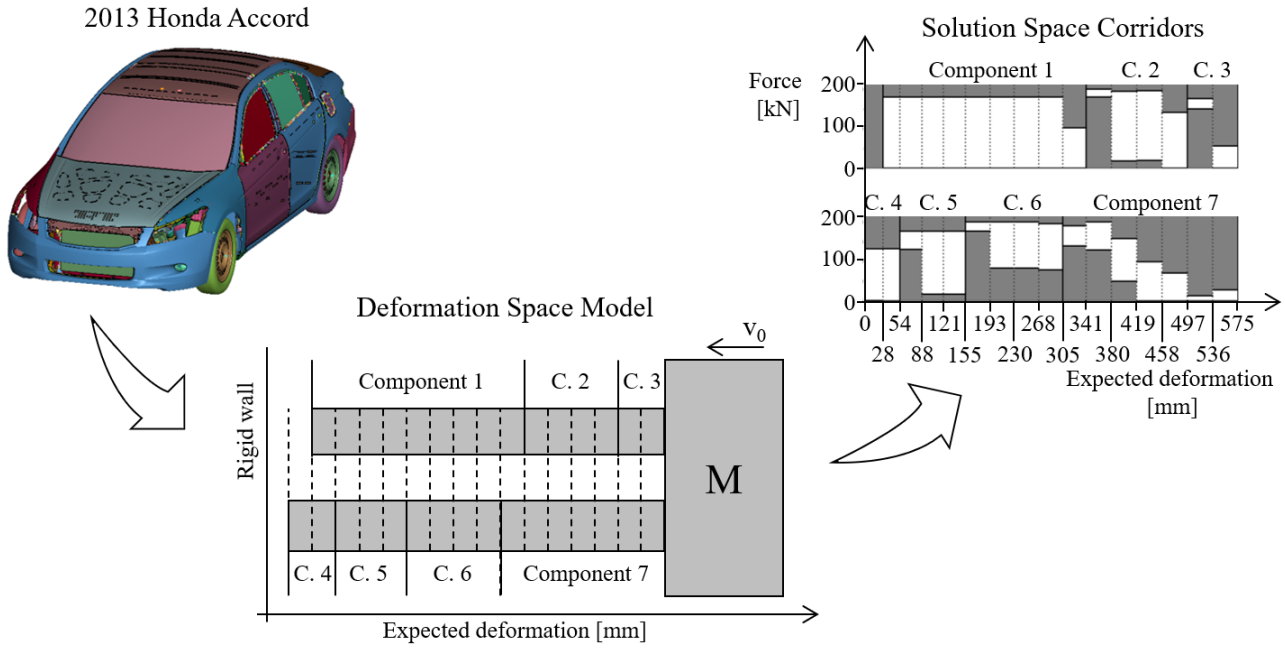


Figure 3: From the 2013 Honda Accord model (National Highway Traffic Safety Administration, b) to the solution space corridors (Daub, 2020), via Deformation Space Model (Lange et al., 2019) of the car considering seven components. In the solution space corridors, the white area represents the feasible force ranges, while the grey area represents the infeasible force ranges.

and deformation fully describes the performance of the vehicle during the frontal impact for our purpose.

To explicit the relation between $\mathbf{F}$ and y, we consider the protocol of the National Highway Traffic Safety Administration (National Highway Traffic Safety Administration, b). The global requirements we are interested in modeling are summarized in the following three conditions:

1. The vehicle must be able to dissipate at least the kinetic energy of an impact at 56 km/h;
2. The maximum deceleration during the impact cannot be higher than 300 m/s^2;
3. The components must deform in a pre-defined order.

These three statements provide the conditions imposed by Daub (2020) on the vector of performance parameters F. The first one, also referred to as *energy constraint*, imposes that the sum of the energy dissipated in each section must be at least equal to the energy possessed by the vehicle. In other terms:

$$\sum_{i=1}^{n} F_i d_i \geq K, \tag{10}$$

where d_i is the deformation length of the section defined in the deformation space model, and F_i is the force level associated with the same section. Finally, K is the kinetic energy the Honda Accord possesses when driving at 56 km/h in the conditions defined in the protocol of the

National Highway Traffic Safety Administration (National Highway Traffic Safety Administration, b).

The second condition, referred to as *impulse constraint*, is related to the deceleration in each section:

$$F_i \leq m_i a_{\max}, \tag{11}$$

where m_i is the active mass of the i-th section, as defined by Daub (2020), and $a_{\max}$ is the maximum allowed deceleration.

Lastly, the force level of the first section of a component must be greater than all force levels of the previous components, in the order in which they should deform. This condition is introduced for reparability. For example, with reference to Fig. 3, Component 2 deforms after Component 1. Consequently, if the maximum force level allowed in Component 1 is 50 kN, then the minimum force level allowed in the first section of Component 2 must be at least 50 kN. The relation $\mathbf{F} \mapsto y$ is, in the end, expressed by a set of equations coming from the conditions reviewed above.

In the last step of the methodology presented by Daub (2020), the solution space Ω_c is computed. In other words, a sub-space of $\mathbb{R}^n$ is defined where all conditions on the vector $\mathbf{F}$ are satisfied and the Cartesian product $\left[F_1^l, F_1^u\right] \times ... \times \left[F_n^l, F_n^u\right]$ can be imposed. All ranges $\left[F_i^l, F_i^u\right]$ are represented in the panel to right of Fig. 3. In this application example, we consider only one component, namely Component 4, a simple crash box.

5.2 Test case: a simple crashbox

As we just said, for the sake of simplicity, we focus only on Component 4, i.e., the crashbox shown in Fig. 4. The local requirements on this component are defined by a maximum deformation length and two corridors (Daub et al., 2020). These quantities are shown in Figure 5. More specifically:

1. the component is allowed to deform of a maximum length $d_{\max} = 54$ mm;
2. the component must absorb a force in between $F_1^l = 0$ kN and $F_1^u = 65.93$ kN while its deformation length ranges from 0 to 28 mm (first section);
3. the component must absorb a force in between $F_2^l = 0$ kN and $F_2^u = 66.02$ kN from 28 to 54 mm of deformation (second section).

According to Section 4, we need to measure from a crash test the mean force over each section, $F_{i,k,\text{msr}}$, and the total deformation length d_{msr}. To do so, we simulate the crash test with a Finite Element model of a drop-tower test of the said component. For the purpose of the simulation, we use the commercial software LS-Dyna (Ansys, Ansys), and the open-lasso-dyna library (open-lasso-dyna, open-lasso-dyna), to interface the solver with the Python coding language. The simulation set-up we use is shown in Fig. 4. The bottom side is clamped, meaning that all

degrees of freedom of nodes on the bottom row are blocked. On the other end of the crashbox, the impactor is modeled as a rigid wall with initial velocity v_0 and mass m. The component itself is then divided into two parts: Part 1 in blue, and Part 2 in green. Each part is linked to a different thickness value of the shell elements, respectively t_1 and t_2. The parameters m, t_1, and t_2 are the set of design parameters we consider as the design variables of our problem. For convenience, we measure the force opposed by the component and the deformation length at the rigid wall. The practice has shown us that, due to the heavy post-processing we apply to the results, measuring the force in a different manner does not affect the end results. When measuring the total deformation length, we account for 1 mm of free fall of the rigid wall. This distance is set in the simulation to avoid any initial penetration, but also to be accounted for in the post-processing of the results.

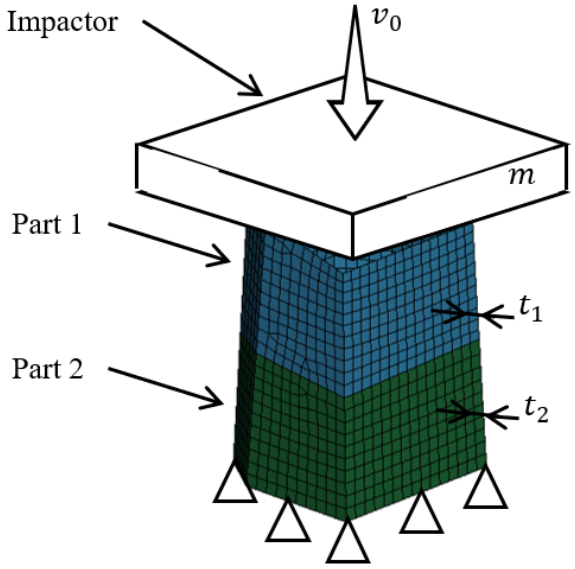


Figure 4: Setup of the droptower test of the studied crashbox.

Always according to Section 4, we define in the simulation a set of design parameters that we can easily control. These are the two thickness values, t_1 and t_2, and the mass of the impactor m. In other words, $\boldsymbol{x} = [t_1, t_2, m]$. We chose these three parameters to guarantee a visible effect when changing the design parameters across their domain.

Lastly, we post-process the results coming from the finite element simulation. In the post-processing, we compute the force levels $F_{i,k,\text{msr}}$, and the total deformation length d_{msr}. To do so, we consider the force-deformation curve measured at the rigid wall. Since for this component there are only two sections, thus two corridors, we compute only two force levels: $F_{1,\text{msr}}$, and $F_{2,\text{msr}}$. The first value is the average force opposed between 0 and 28 mm of deformation. The second, instead, is the average force opposed between 28 and 54 mm of deformation. $F_{1,\text{msr}}$, $F_{2,\text{msr}}$, and d_{msr} are then used to compute the limit state function $g(\boldsymbol{x})$.

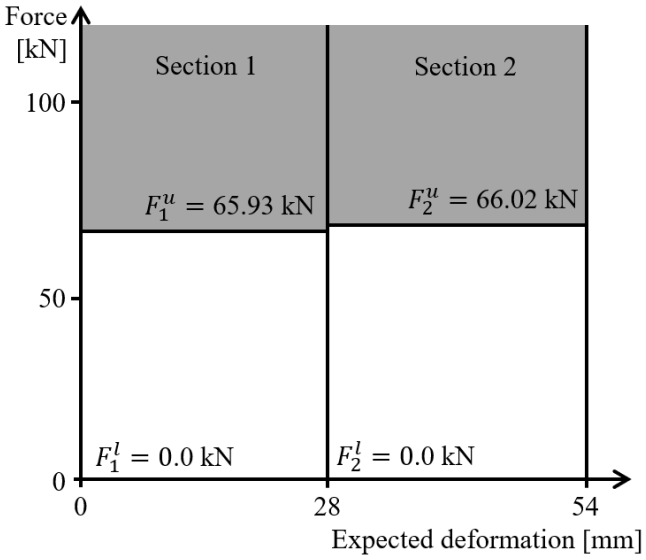


Figure 5: Corridors of the crashbox (Daub, 2020).

5.3 Definition of the limit state function

As introduced in Section 4, the limit state function of the crashbox example is defined as the minimum of three sub-functions $h_i(\boldsymbol{x})$:

$$g(\boldsymbol{x}) = \min\left(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), h_3(\boldsymbol{x})\right). \tag{12}$$

The first and second functions are related to the two corridors. Thus they are defined according to Eqn. (8):

$$h_1(\boldsymbol{x}) = \min\left(F_{1,\mathrm{msr}-F_1^l}, F_1^u - F_{1,\mathrm{msr}}\right), \tag{13}$$

$$h_2(\boldsymbol{x}) = \min\left(F_{2,\mathrm{msr}-F_2^l}, F_1^u - F_{2,\mathrm{msr}}\right). \tag{14}$$

The last function $h_3(\boldsymbol{x})$ expresses the condition on the maximum displacement. Therefore, it is formulated according to Eqn. (9):

$$h_3(\boldsymbol{x}) = \min\left(d_{\max} - d_{\mathrm{msr}}\right). \tag{15}$$

The limit state function we just defined is *positive* if $0 \leq F_{1,\mathrm{msr}} \leq 65.93$ kN, $0 \leq F_{2,\mathrm{msr}} \leq 66.02$ kN, and $d_{\mathrm{msr}} \leq 54$ mm. If at least one of these conditions is violated, then $g(\boldsymbol{x})$ is negative.

Having defined a limit state function, we can apply the ALR algorithm to learn an approximation $\widehat{g}(\boldsymbol{x})$ of $g(\boldsymbol{x})$ that is accurate in sign. To do so, we use the implementation of the ALR algorithm available in the UQLab software (Marelli and Sudret, 2014; Moustapha et al., 2022b). The algorithm provides several options for the surrogate model, the enrichment strategy, and the stopping criteria. Following the recommendations of Moustapha et al. (2022a), we use polynomial-chaos-based Kriging as a surrogate model, and subset simulation as the reliability estimation method. We start with an initial experimental design of 20 samples. As a stopping criterion, we simply use a fixed budget of 250 additional finite element simulations, resulting in a total of

270 model runs. Because the ALR algorithm requires a probabilistic description of the input parameters, we provide the three design parameters t_1, t_2, and m as uniform random variables, following the recommendation in Abbiati et al. (2022). This choice avoids creating *a-priori* preferential regions in the design space, allowing for an even exploration by ALR. While t_1 and t_2 range between 0.7 and 4.0 mm, the mass m ranges from 20 to 200 kg. These ranges are arbitrarily chosen, yet they lead to reasonable results for such a crashbox. For clarity, the variables and their properties are summarized in detail in Tab. 1, while the detailed settings used in the ALR algorithm are summarized in Tab. 2.

Table 1: Summary of the design variables of the crashbox

Name	**Type**	**Range**
t_1	Uniform	$[0.7\text{mm}, 4.0\text{mm}]$
t_2	Uniform	$[0.7\text{mm}, 4.0\text{mm}]$
m	Uniform	$[20\text{kg}, 200\text{kg}]$

Table 2: Summary of the options used in the ALR algorithm

ALR Setting	**Option**
Surrogate Model Function	Polynomial-Chaos-Based Kriging
Reliability Estimator	Subset simulation
Stopping Criteria	Maximum number of evaluations: 270
Size Initial Data-Set	20 samples
Number Added Samples	250 samples

After completing the algorithm, $\widehat{g}(\boldsymbol{x})$ is trained on the 270 samples, of which 20 initial samples are sampled with a Latin Hypercube sampling strategy (LHS, (McKay et al., 1979)), and 250 samples are actively placed by the ALR algorithm. We can now use $\widehat{g}(\boldsymbol{x})$ as a classifier to predict the crashworthiness of a design. According to the definition of $g(\boldsymbol{x})$, a design is classified as crashworthy when $\widehat{g}(\boldsymbol{x}) \geq 0$ and it is unworthy when $\widehat{g}(\boldsymbol{x}) \leq 0$.

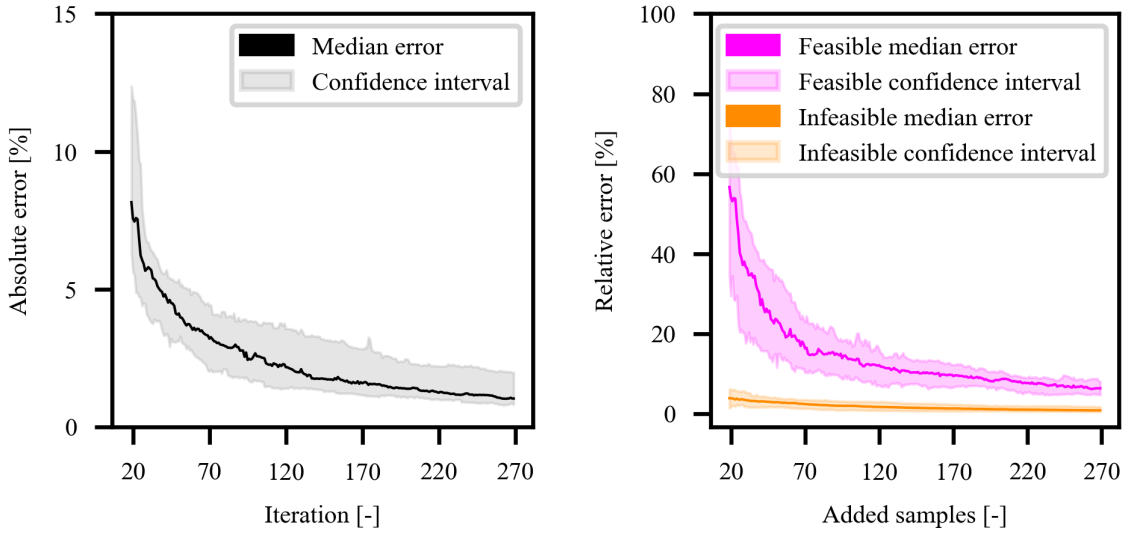
5.4 Results

To assess the robustness of the method, we repeat the training $N_\text{R} = 30$ times, each time changing the initial random seed to capture its sensitivity to its initialization. For the sake of validation, we also generate a large out-of-sample validation set, consisting of $N_\text{Val} = 10,000$ full LS-Dyna simulations, post-processed to evaluate the quantities $F_{1,\text{msr}}$, $F_{2,\text{msr}}$, and d_msr, thus $g(\boldsymbol{x})$. This validation set is generated using LHS. Each of the N_R replications is compared against this reference set to estimate meaningful accuracy metrics.

For each training run, we track three performance indicators: the percentage of misclassified samples, the percentage of samples classified as feasible that are misclassified — hereinafter

referred to as *feasible relative error* —, and the percentage of samples classified as infeasible that are misclassified — hereinafter referred to as *infeasible relative error*. The evolution of the first index is shown in Fig. 6a, in which the solid line marks the average percentage of samples misclassified — i.e. the median total error of the classifier — over the 30 runs. The total error is consistently decreasing during training and converges around 1.04 %. The shaded area represents the empirical 10-90% confidence bounds on the total error, based on the available replications. The confidence interval settles around a value of 1.17 %. In this application, we observe a maximum misclassification of 2.86 %. The overall performance of the classifier for this application is also reported in the confusion matrix of Fig. 7. In this matrix, we report the absolute and relative mean and standard deviation of the number of misclassified samples. To compute the relative number, we consider the ratio between the prediction and the number of samples of the true class. The matrix confirms the good performance of the classifier, with the majority of the validation samples being correctly classified.

In Fig. 6b, the purple curve and shaded area represent the median feasible relative error and its empirical 10-90 % confidence interval over the 30 runs. On average, this relative error settles at 6.39 %, while its confidence interval at 3.45 %. The orange curve and shaded area in the figure, instead, represent the infeasible relative error (respectively median and 10-90 % confidence interval). The median error converges to 0.87 %, with a standard deviation of 1.25 %.



(a) (b)

Figure 6: Convergence of the percentage of misclassified samples: (a) the percentage of the 10,000 samples misclassified by the trained classifiers; (b) the percentage of the samples classified as feasible that have been misclassified in magenta, and the percentage of the samples classified as infeasible that have been misclassified in orange. The solid line shows the median value over the 30 repetitions, the shaded area shows the 10-90 % confidence interval.

Of the 30 runs, we show more in detail the one with a total error of 1.02 % — i.e. it misclassifies

		True class	
		Feasible $N_F = 699$	Infeasible $N_I = 9303$
Predicted class	Feasible	617 ± 52 $\frac{N}{N_F} = 0.883 \pm 0.074$	42 ± 9 $\frac{N}{N_I} = 0.005 \pm 0.001$
	Infeasible	82 ± 52 $\frac{N}{N_F} = 0.117 \pm 0.074$	9260 ± 10 $\frac{N}{N_I} = 0.995 \pm 0.001$

Figure 7: Confusion matrix of the trained classifier with the median performance.

a total of 102 samples. Of these 102 samples, 48 are samples wrongly classified as feasible, and 54 are wrongly classified as infeasible. Therefore, the relative feasible error of this run is 6.93%, and the relative infeasible error is 0.58%. Since the domain of the design parameters is three-dimensional, we can visualize $\mathcal{D}_F$ and $\mathcal{D}_S$, i.e. the crashworthy domain and the unworthy one. To do so, we slice the domain by fixing each design parameter at a time and plotting a slice at the fixed value. Figure 8 shows the comparison between the estimated feasible areas (in magenta) and the actual feasible areas (in grey) obtained from the validation set. To obtain the reference shaded areas in grey we linearly interpolate between the values of $g(\boldsymbol{x})$ of the 10,000 validation samples evaluated with LS-Dyna. Similarly, to obtain the estimated feasible areas on the given slice, we interpolate between the 10,000 validation samples evaluated on the approximation $\widehat{g}(\boldsymbol{x})$ returned by the ALR algorithm. This figure makes use of the converged classifier and allows us to understand how well the trained classifier is approximating the feasible areas. The slices are placed in areas of interest. For t_1 and t_2, two slices are placed close to the borders of the domain and two are placed in the middle. For m, the four slices are placed in the lower part of the domain (lower than 50 kg). The rest of the domain is mainly empty, hence, not of interest. This figure allows us to assess how the classifier estimates the feasible areas in the design domain after a total training cost of $N = 270$ samples. By interpolating between the 10,000 samples of the validation set, we can visualize the complicated topology of the feasible area, its small volume (approximately 7% of the design space), and the areas where the classifier accuracy is reduced. The real and estimated feasible areas are mostly overlapping. However, the classifier starts to show inaccuracies where the true feasible area has many small isolated peaks, around $m = 40$ kg and $m = 50$ kg. Please note that a quantitative interpretation is difficult because of the limited data available for interpolation. As shown in Fig. 6, the overall accuracy of the classifier increases with the available computational budget, and we, therefore, expect that even these areas would be better reconstructed with further iterations of the algorithm.

Finally, Fig. 9 provides a comparison between the feasible area estimated by the ALR algorithm and the feasible area defined by the validation set of 10,000 simulations throughout the training of the classifier. In this figure, we show one slice per dimension per training snapshot. Therefore,

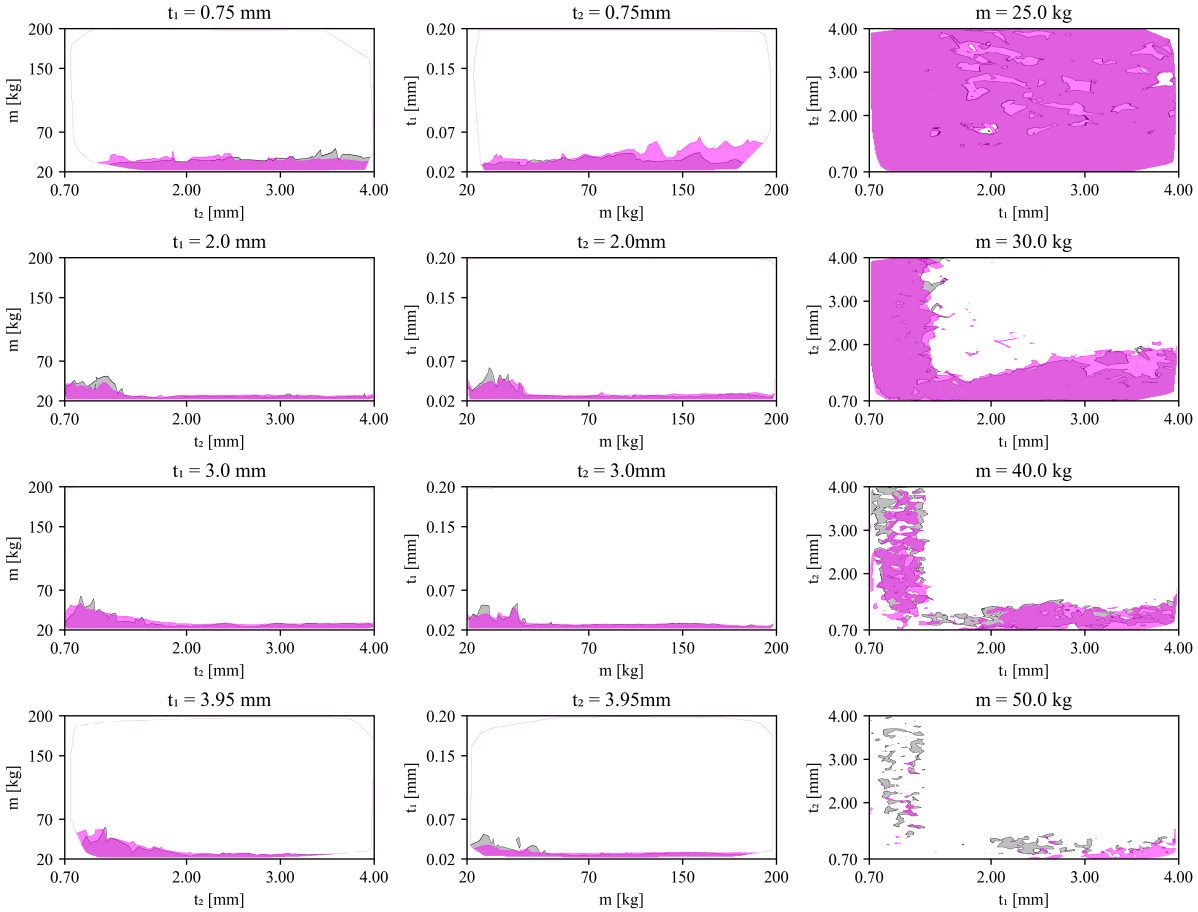


Figure 8: Comparison between the feasible areas estimated by the trained classifier and defined by the validation set of 10,000 simulations. The grey area is the feasible area according to the validation set, the magenta area is the estimated feasible area. The 12 slices are taken for the following values: $t_1 = 0.75, 2.0, 3.0, 3.95$ mm, $t_2 = 0.75, 2.0, 3.0, 3.95$ mm, and $m = 25, 30, 40, 50$ kg.

the slices are placed at $t_1 = 1.0$ mm, at $t_2 = 1.0$ mm, and the last at $m = 30$ kg. The snapshots of the training are taken with 50, 100, 200, and 270 samples: in other words, towards the beginning of the training, in the middle of the training, and at convergence. The areas marked in grey are the feasible areas according to the validation set. The areas in magenta are the feasible areas according to the trained classifier. This figure allows us to compare the validation set with the trained classifier at different stages of training to understand how the algorithm behaves. For more detailed plots, the reader is referred to the Appendix.

6 Discussion

The results from the practical application presented in Section 5.4 indicate that the methodology we are proposing efficiently and effectively trains a classifier capable of distinguishing crashworthy

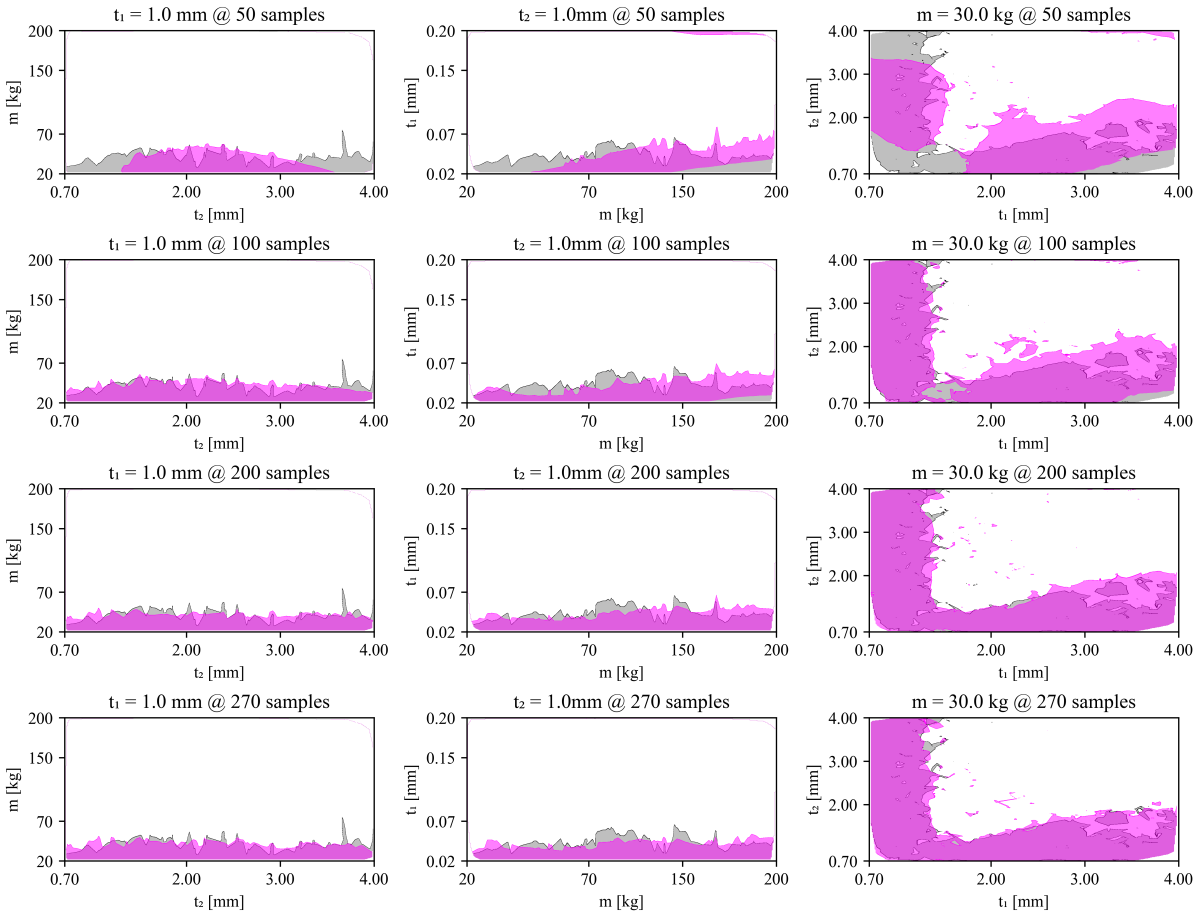


Figure 9: Evolution of the estimated feasible area during training. Each row shows three slices placed at $t_1 = 1.0$ mm, $t_2 = 1.0$ mm, and $m = 30$ kg. The first row shows the estimated feasible area after the evaluation of 50 samples (20 points of initial DoE and 30 iterations). The second row shows the said area after 100 samples (20 points of initial DoE and 80 iterations). The third row shows the area estimated with 200 samples (20 points of initial DoE and 180 iterations). The last row shows the feasible area at convergence, i.e. after having evaluated 270 samples. The grey area shows the feasible areas according to the validation set. The magenta areas, instead, show the feasible areas estimated by the trained classifier.

designs from non-crashworthy ones. With just 270 samples, we are capable of training a classifier with a median total error of 1.04 %.

The fact of having a median total error of 1.02 %, despite the small volume of the feasible domain, is possible due to the active learning strategy. This type of learning places most samples close to areas generally treated as of high interest, according to the learning functions defined in Section 3. Figure 6 shows that the chosen learning strategy is very effective: the algorithm quickly identifies the feasible area and the estimation error quickly drops to a small value. The ability to locate samples “smartly" increases the accuracy of the classifier while keeping low the training effort. In other words, we spend computational power only where needed, namely at

the boundaries of the crashworthy areas. This contrasts with the more common technique of building a general surrogate of the whole finite element simulation and using its output to test the feasibility against a set of local requirements (Amsallem et al., 2012).

The effectiveness of the ALR tools is also evident in Fig. 9. The plots in the first row show that 50 samples are not enough to properly approximate the feasible areas. However, as the second row indicates, 100 samples already yield a good approximation of these areas. At this stage, the total error is already well below 5 %. The majority of this error is attributed to the fact that there are feasible areas with complicated topology. For example, an area that requires more attention is around the point with $t_1 = 1.0$ mm, $t_2 = 0.8$ mm, and $m = 0.03$ ton, on the left bottom corner of the third plot on the second row of Fig. 8. This fact indicates that the algorithm tends to behave conservatively, by classifying crashworthy samples as not-crashworthy, rather than vice versa. From a practical point of view, this is in favor of a more robust development: the algorithm first identifies big areas of success, and then slowly expands them to refine the approximation.

More evidence of the conservativeness of the algorithm is found in Fig. 6. While Fig. 6a indicates that the classifier becomes more and more precise with each iteration, the plot in Fig. 6b indicates that most of the improvements are made on reducing the number of crashworthy samples misclassified, and thus the feasible relative error. The fact that the feasible relative error is always bigger than the infeasible relative error supports the conservativeness of the algorithm. On top of this, both relative errors remain relatively stable after the 200th evaluation. At this stage, the total error is still slightly decreasing. Therefore, we can infer that in this application, on average, by iteration 200 the algorithm has identified all feasible areas.

In this application, the algorithm starts refining the classifier after the 200th iteration. It does so by identifying small feasible areas encapsulated in infeasible zones, and vice-versa. The result of this refinement can be seen in the fully converged plots of Fig. 8 for $m = 25$ kg. In this plot, the algorithm was capable of identifying some of the small infeasible areas that are completely surrounded by feasible zones. However, notice that letting the algorithm fully refine these areas provides only a limited benefit compared to the computational effort required. These areas are small enough to not deeply affect the total error.

The relatively simple system selected to apply the proposed methodology, namely a crashbox where we vary impactor mass and the side wall thicknesses, is done on purpose. By choosing such a well-known case study, we could understand better the very mechanisms of active learning in this new context. More challenging industrial case studies should be tackled in the near future.

In conclusion, our proposed method for training a crashworthiness classifier is efficient, accurate, and appears conservative in the selected case study. It is efficient because it uses a limited number of evaluations to identify the crashworthiness domain — with 200 samples, the classifier prediction already has an error lower than 5 % according to Fig. 6a. It is accurate because it converges with only 270 evaluations around a total error of only 1.04 %. Finally, it is conservative because it favors misclassifying crashworthy samples, rather than misclassifying failing samples.

7 Conclusion

The methodology we present in this paper merges the Solution Space Method (SSM) with Active Learning Reliability (ALR). This coupling provides an efficient and effective approach for addressing some of the challenges in finding crashworthy designs of components for new vehicles. By cascading global requirements via the SSM on a local level and utilizing ALR to propagate the cascaded local requirements on parametrized components, our approach provides a classifier capable of identifying possible crashworthy designs of the given component. The presented simple application — a crashbox with three design parameters — demonstrates the efficiency, accuracy, and conservativeness of the methodology. With a rate of only 2.5 % of misclassification, we successfully train a crashworthiness classifier evaluating a total of 270 samples.

Our methodology should serve as a valuable new tool for designers and engineers. It offers an effective method to identify potential feasible designs at a low budget, thus unlocking the full potential of any given design. The positive outcomes of this study allow us to underline the importance of coupling systems engineering methods with machine learning to better exploit the information they provide and, thus, better support the development of new products.

Acknowledgments

The authors would like to thank Adéla Hlobilová for her invaluable support in the use of the UQ[Py]Lab software.

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 955393.

References

Abbiati, G., S. Marelli, C. Ligeikis, R. Christenson, and B. Stojadinović (2022). Training of a classifier for structural component failure based on hybrid simulation and kriging. *Journal of Engineering Mechanics 148*(1), 04021137.

Amsallem, D., M. J. Zahr, and C. Farhat (2012). Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering 92*(10), 891–916.

Ansys, I. Ls-dyna: Crash simulation software. `https://www.ansys.com/products/structures/ansys-ls-dyna`, accessed June 25, 2024.

Daub, M. (2020). *Optimizing flexibility for component design in systems engineering under epistemic uncertainty.* Ph. D. thesis, Technical University of Munich, Munich, Germany.

Daub, M., F. Duddeck, and M. Zimmermann (2020). Optimizing component solution spaces for systems design. *Structural and Multidisciplinary Optimization 61*(5), 2097–2109.

Ditlevsen, O. and H. O. Madsen (1996). *Structural reliability methods*, Volume 178. John Wiley & Sons Ltd.

Fender, J. (2014). *Solution spaces for vehicle crash design.* Ph. D. thesis, Technical University of Munich, Munich, Germany.

Fender, J., F. Duddeck, and M. Zimmermann (2017). Direct computation of solution spaces. *Structural and Multidisciplinary Optimization 55*(5), 1787–1796.

Graf, W., M. Götz, and M. Kaliske (2018). Computing permissible design spaces under consideration of functional responses. *Advances in Engineering Software 117*, 95–106.

Graff, L. (2013). *A stochastic algorithm for the identification of solution spaces in high-dimensional design spaces.* Ph. D. thesis, University of Basel, Basel, Switzerland.

Graff, L., H. Harbrecht, and M. Zimmermann (2016). On the computation of solution spaces in high dimensions. *Structural and Multidisciplinary Optimization 54*, 811–829.

Krischer, L. and M. Zimmermann (2021). Decomposition and optimization of linear structures using meta models. *Structural and Multidisciplinary Optimization 64*(4), 2393–2407.

Kwok, J.-Y. and I.-H. Tsang (2004). The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks 15*(6), 1517–1525.

Lange, V. A., J. Fender, L. Song, and F. Duddeck (2019). Early phase modeling of frontal impacts for crashworthiness: from lumped mass-spring models to deformation space models. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 233*(12), 3000–3015.

Lemaire, M., A. Chateauneuf, and J.-C. Mitteau (2009). *Structural reliability.* John Wiley & Sons, Ltd.

Marelli, S. and B. Sudret (2014). Uqlab: A framework for uncertainty quantification in matlab. In *Vulnerability, Uncertainty, and Risk (Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom)*, pp. 2554–2563. American Society of Civil Engineers.

McKay, M. D., R. J. Beckman, and W. J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 239–245.

Moustapha, M., S. Marelli, and B. Sudret (2022a). Active learning for structural reliability: Survey, general framework and benchmark. *Structural Safety 96*, 102174.

Moustapha, M., S. Marelli, and B. Sudret (2022b). Uqlab user manual – active learning reliability. *Report UQLab-V2, in Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich*.

Moustapha, M. and B. Sudret (2019). Surrogate-assisted reliability-based design optimization: a survey and a unified modular framework. *Structural and Multidisciplinary Optimization 60*(5), 2157–2176.

National Highway Traffic Safety Administration. Crash simulation vehicle models. `https://www.nhtsa.gov/crash-simulation-vehicle-models`, accessed June 25, 2024.

National Highway Traffic Safety Administration. Laws and regulation. `https://www.nhtsa.gov/laws-regulations`, accessed June 25, 2024.

open-lasso-dyna. lasso-python. `https://github.com/open-lasso-python/lasso-python`, accessed June 25, 2024.

Sauder, T., S. Marelli, and A. J. Sørensen (2019). Probabilistic robust design of control systems for high-fidelity cyber–physical testing. *Automatica 101*, 111–119.

Teixeira, R., M. Nogal, and A. O'Connor (2021). Adaptive approaches in metamodel-based reliability analysis: A review. *Structural Safety 89*, 102019.

Vogt, M. E., F. Duddeck, M. Wahle, and M. Zimmermann (2019). Optimizing tolerance to uncertainty in systems design with early-and late-decision variables. *IMA Journal of Management Mathematics 30*(3), 269–280.

Zimmermann, M. and O. De Weck (2021). Formulating engineering systems requirements. In *Handbook of Engineering Systems Design*, pp. 1–49. Springer.

Zimmermann, M., S. Königs, C. Niemeyer, J. Fender, C. Zeherbauer, R. Vitale, and M. Wahle (2017). On the design of large systems subject to uncertainty. *28*(4), 233–254.

Zimmermann, M. and J. E. Von Hoessle (2013). Computing solution spaces for robust design. *International Journal for Numerical Methods in Engineering 94*(3), 290–307.