Journal of Mathematics in Industry
a SpringerOpen Journal

RESEARCH Open Access

Increasing design flexibility by manually adapting the solution space for crashworthiness

Paolo Ascia[1*], Volker A. Lange[2] and Fabian Duddeck[1]

*Correspondence: paolo.ascia@tum.de
[1]TUM School of Engineering and Design, Technical University of Munich, Arcisstr. 21, 80333, Munich, Germany
Full list of author information is available at the end of the article

Abstract
The solution space methodology, as presented in 2013, was meant to guide developers at the very beginning of the development process of a new mechanically crashworthy car. Several attempts were already made to use this methodology at later development stages. However, they all encountered problems related to its very strict and demanding corridors, thus constricting the design parameters. To allow more flexibility, two different approaches were proposed to relax the initial strict conditions. The first introduced temporal dependencies to widen the corridors. The second locally changed the corridors to adapt to the needs of the development, introducing dependencies between components. We, on the contrary, propose a new method to increase flexibility without introducing any kind of dependencies. We manage this by computing the intervals of solution space under user-defined conditions, hence selecting a custom set of independent corridors that fits the data gathered during development; i.e.: force-deformation curves that can be measured during a drop-tower test simulation. This new methodology of the adaptive solution space allows designers to edit the corridors, in order to have more flexibility for fulfilling high-level requirements when independently designing new components.

Keywords: Design for flexibility; Solution spaces; Systems engineering; Concurrent engineering; Crashworthiness

1 Introduction

Designing a new car came to be more and more demanding: vehicles must fulfil an increasing number of requirements, coming both from governmental agencies, like those for road safety, or for a reduced fuel consumption, and the desires of the customers, such as the connectivity with their smart devices or the driving comfort. Thus, manufacturers have taken advantage of the methodologies of systems engineering to cope with the new challenges and help themselves to better plan the development from the concept to the final product. We focus on the solution space methodology proposed in [18]. This method was inspired by the V-model concept [10] of cascading high-level requirements to a component level. The method allows the development of different components and testing their requirement-fulfilment independently. Here, we consider only the application of solution spaces that involves the development of components for mechanical crashworthiness.

The solution space method cascades high-level requirements, like those defined in the legislation, to low-level ones. By low-level requirements, we mean the performance a component needs to fulfil so that the final assembly meets the high-level requirements. This method is meant for the very early stages of development when only little information is available for any of the components to develop. The low level-requirements, once computed, are not intended to change over time. However, during the development, the information gained reveals unforeseen challenges and difficulties. Thus, on one hand, the rigidity of the solution space method limits the ability to change and learn from the new situation. On the other hand, the way it cascades the requirements gives the developers the chance to develop different components simultaneously. Consequently, different components can be developed at the same time and their assembly will meet the high-level requirements. To better demonstrate how this dispute between perceived rigidity of the low level requirements and the independence in simultaneous design manifests in the practice, let us introduce an application in vehicle design for crashworthiness.

The solution space methodology is based on an analytical model. The latter is used to evaluate what effects the high-level requirements have on a component level. We focus on the application for the mechanical crashworthiness during the full rigid barrier test defined in [14]. Therefore, the analytical model represents the crash itself, where the mathematical variables capture the forces absorbed by each component when deforming. The high-level requirements are defined by the protocols published by governmental agencies or consumer protection — generally available on their websites, like [6, 14]. The solution space methodology, then, translates these high-level requirements into a set of independent ranges — called corridors — for each variable, representing the to-fulfil performance matching to the overall conditions of crashworthiness.

Although the solution space methodology is a powerful tool for decoupling the design of crashworthy components, literature shows that the method can request a performance not physically achievable. Consider the work in [1]; the author attempted to optimize the thickness of some components to reduce their mass, using the corridors of solution space as constraints. The optimization, however, could not find a feasible solution for all components. To handle this situation the work of [4, 17] suggests introducing temporal dependencies between variables. It distinguishes between components that can be fixed at the beginning because of a limited physical performance, and those that are not limited, and hence can be designed later. Another approach is to impose local dependencies to edit the corridors, like the work in [15]. This approach evaluates the underlying relation between components and allows to change the corridors of each variable until the high-level requirements are satisfied locally.

These variations of the solution space method relax the initial condition of independence to make it more flexible. This avoids to run into the previously mentioned dispute between rigid low level requirements and independence in concurrent design. However, it causes the methodology to lose the ability to develop all components in parallel. We want a way to increase the flexibility of the solution space method, without limiting its potential: concurrent and crashworthy design of all components throughout the whole development. Such flexibility enables tailoring the solution space to the problems of the development, while still allowing the development of all components at the same time. In this paper, we propose a new method to edit the corridors representing the solution space: we call it adaptive solution space.

The adaptive solution space methodology we propose is meant to allow more flexibility and maintain both properties of the solution space after the development has started. This means that we suppose some development-related information is available for at least one of the new components. The adaptive method re-computes the solution space under a set of constraints defined by the developers. These constraints capture the problems developers are facing when trying to fulfil the initial corridors; for example, a component not fulfilling the corridors, like the situation showcased in the application of Sect. 5 in Fig. 9. Our method takes the information just gained during the first iteration of the design and re-computes an independent solution space to adapt to the challenges encountered.

The paper is structured to first offer an overview of the current state of the art of the solution space methodology and exemplify it with a simple test case, to better showcase the problem at hand. Then, we introduce our proposed method: adaptive solution space. The last sections present an example of the application of the proposed method, to help the reader understand how it can be used.

2 Solution space

The solution space methodology was first introduced as a systems engineering method to support product development [18]. This approach assigns to any design parameter an interval, to represent the constraints imposed by high-level requirements. The choice of these sets is not arbitrary: they ensure feasibility, independence and flexibility. The first means that, if the design parameter is within the given set of intervals, the design will fulfil all high-level requirements — in other words, the new design guarantees the crashworthiness of the car.

The second property, independence, allows the simultaneous development of all design parameters; for example, if each parameter corresponds to a crash-relevant component, these can be developed independently of each other. When assembled, no further changes are needed to these parts. Lastly, the flexibility property translates into having the widest possible intervals. This allows the maximum freedom of choice and multiple possible realizations.

To compute the solution spaces M. Zimmermann et al. proposed in [18] two methods: the direct and the indirect method. The indirect method is based on sampling and clustering [5, 7, 9]. The direct method is based on an analytical approach [2, 7, 8]. We focus on this second one.

2.1 Direct method for solution space

The direct method is based on an analytical approach that computes the set of intervals assigned to the design parameters based on a low-fidelity model. Therefore, this method aims at linking the design parameters – the input – to the said intervals – the output.

The first step is to build a low-fidelity model of the vehicle structure [7]. Such a low-fidelity model can be the Geometry Space Model and the Deformation Space Model [12]. Through them, the design parameters are represented and linked to their performance. Let us group the design parameters in the vector $\boldsymbol{F}$, as proposed in [3]. For each parameter an upper and lower limit is defined, thus delimiting the design space Ω_{ds}. These limits can be arbitrarily set; it is generally left to the experience and knowledge of the developers. Since the search for the solution space is performed within Ω_{ds}, they also limit the solution space itself. To help the reader understand, consider Fig. 1. It represents the solution space

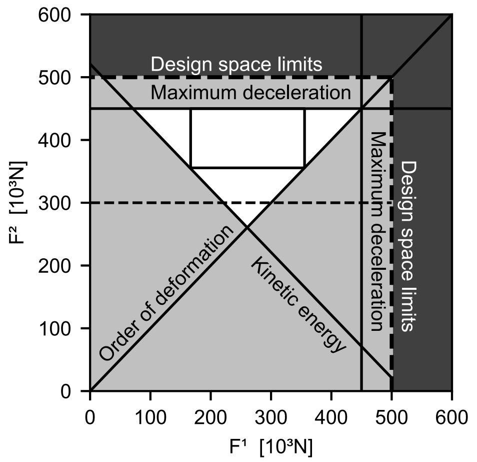

Figure 1 Representation of the two-dimensional design space: the light grey area represents the design space, the white area — delimited by the maximum deceleration, the order of deformation and the minimum energy absorption constraints — represents the space of feasible solutions, and the area inside the rectangle is the area of feasible and independent solutions. The dashed line corresponding to $F_2 = 300$ kN represents the desired design points the developers want to include in the solution space

of the example described in the next section. The limits are represented in this figure by the dashed lines marked as *design space limit*. Therefore, the search for the solution space is limited to the region in light grey.

Having defined the input, we can look at the link with the output $\boldsymbol{R}$ — the set of intervals. $\boldsymbol{R}$ represent the cascade of the high-level constraints. Therefore, between input and output there must be a function capable of representing the constraints of the high-level requirements. We call this function *performance function*, f_{perf}. It generally consists of a set of inequalities. For example, in Fig. 1, the constraints of the performance function are represented by the lines marked as *Maximum deceleration*, *Kinetic energy*, and *Order of deformation*. The white area, then, represents the feasible space Ω_{c}, where all conditions are fulfilled, while the grey area is the one where at least one condition is not fulfilled. Mathematically, the relation between the input $\boldsymbol{F}$ and the output $\boldsymbol{R}$ is expressed by:

$$\Omega_{\mathrm{c}} = \left\{\boldsymbol{F} \in \Omega_{\mathrm{ds}} \mid f_{\mathrm{perf}}(\boldsymbol{F}) \leq \boldsymbol{R}\right\} \subseteq \Omega_{\mathrm{ds}}. \tag{1}$$

So far, Ω_{c} guarantees only one of the three properties of the solution space: feasibility. The second property, independence between design parameters, is obtained by imposing that the set of intervals must form an axis-parallel base of a subset of Ω_{c}:

$$R^1 \times R^2 \times \cdots \times R^n \in \Omega_{\mathrm{c}}. \tag{2}$$

Where the i-th interval can be expressed by its upper and lower bound, i.e. $R^i = [R^i_{\mathrm{L}}, R^i_{\mathrm{U}}]$. In practice, this condition reduces the feasible subspace to a hypercube fitted inside Ω_{c} itself.

Lastly, to allow maximum flexibility we need to make sure the subset of Ω_{c} is as big as possible. To do so, we maximize the volume contained in the subset described in Equation (2):

$$\begin{aligned} &\max \quad \prod_{i=1}^{n} \left[R_{\mathrm{L}}^{i} - R_{\mathrm{U}}^{i}\right] \\ &\text{s.t.} \quad \left[R_{\mathrm{L}}^{1}, R_{\mathrm{U}}^{1}\right] \times \left[R_{\mathrm{L}}^{2}, R_{\mathrm{U}}^{2}\right] \times \cdots \times \left[R_{\mathrm{L}}^{n}, R_{\mathrm{U}}^{n}\right] \subset \Omega_{\mathrm{c}}. \end{aligned} \tag{3}$$

Notice that the subspace is convex, as explained in [2, 11]. The solution to the problem of Equation (3) is computed with the interior point methods available in [16]. The solution space found in this way — called Component Solution Space in [2] — provides a set of intervals per design parameter that ensures the permitted values to be all feasible and independent of each other. They are feasible because they are all contained in Ω_{c}; they are independent of each other because they form an orthogonal base thanks to the relation in Equation (2). The intervals so computed are called corridors. By fulfilling the optimization problem in Equation (3), the corridors allow the maximum flexibility.

2.2 Example of computing solution space with the direct method

To better understand how the Component Solution Space is computed, we present in this section a simple example. This very same example was used in [3]. We will also refer to this example later in the paper to showcase how the proposed adaptive solution space method works. The purpose of this example is to visualize the different subspaces, thus, it is two-dimensional. Therefore, the entire approach is built around only two variables so that the space they are describing is easily visualized on a plane. It is set up with the information taken from the open-source model of the Honda Accord [13]. Moreover, it is built for the front full rigid barrier test [14].

Following the step presented earlier in Sect. 2.1, we first need to define the Geometry Space Model. We consider the front of the car to be made out of two identical components, positioned in series on the same load-path. This Geometry Space Model is shown in Fig. 2, where the black part represents the undeformable length (behaving like a rigid body) and the grey one represents the deformable length. The mass M, instead, accounts for the mass of the vehicle behind the firewall. In the second step, we define the Deformation Space Model: for the sake of simplicity, there are only two sections, one per component. The model is shown in Fig. 3, where we can see that each component is supposed to deform

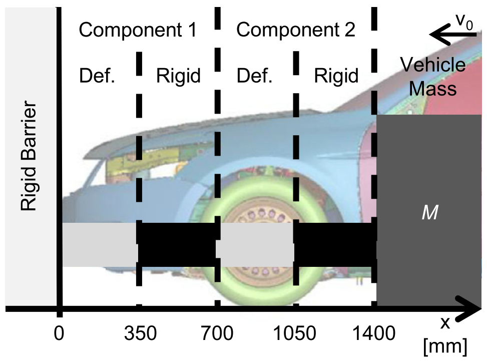

Figure 2 Geometry Space Model with two components of the Honda Accord Model [13], according to the procedure described by [12]

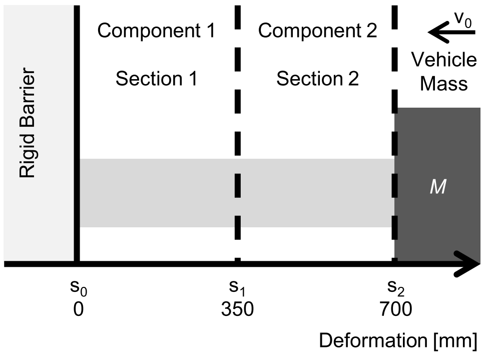

Figure 3 Deformation Space Model with two components of the Honda Accord Model [13], according to the procedure described by [12]

350 mm. The variables of the model of the solution space are, then, the force levels F^1 and F^2, respectively describing section 1 and section 2. However, to build the model itself we need the thresholds values F_c and the performance function f_perf for both F^1 and F^2. These are taken from the protocol for the full rigid barrier test [14]:

1 Maximum deceleration during the impact of 300 m/s^2;
2 Initial impact velocity of 56 km/h;
3 Order of Deformation — the components deform in a specific order, so that the first component to deform is easy to repair, whereas the last one is the most difficult to repair.

These conditions translate to the following three physical relations; respectively:

1 Maximum deceleration: $\frac{F^i}{m_\mathrm{active}} = a^i \le a_\mathrm{max} = 300\ \mathrm{m/s}^2$;
2 Initial impact velocity: $F \cdot s \le \frac{1}{2} m v_0^2$;
3 Order of Deformation: $F^i \le F^{i+1}$;

With reference to the i-th section, F^i is the absorbed force, m_active is the active mass — the mass behind the i-th section being decelerated — a^i is the deceleration, a_max is the maximum deceleration allowed during impact, v_0 is the initial impact velocity, and F^{i+1} is the force absorbed by the next section, less accessible according to the order of deformation. Finally, F is the sum of all F^i, and s is the total deformation length.

Let us now substitute the variables in the equations and explicit f_perf and $\boldsymbol{F}_c$. We consider a constant active mass of 1500 kg. This yields for Ω_c the following system of inequalities:

$$\Omega_\mathrm{c} : \begin{cases} F^1 \le 450\ \mathrm{N}; & \text{(maximum deceleration)}, \\ F^2 \le 450\ \mathrm{N}; & \text{(maximum deceleration)}, \\ 0.23F^1 + 0.23F^2 \le -121\ \mathrm{N}; & \text{(initial impact velocity)}, \\ F^1 - F^2 \le 0; & \text{(order of deformation)}. \end{cases} \tag{4}$$

The feasible subspace is represented by the white area in Fig. 1. It follows that the Equation (3) is reformulated as follows:

$$\begin{aligned} \max \quad & \prod_{i=1}^{2} \left[F_\mathrm{L}^i - F_\mathrm{U}^i\right] \\ \text{s.t.} \quad & \left[F_\mathrm{L}^1, F_\mathrm{U}^1\right] \times \left[F_\mathrm{L}^2, F_\mathrm{U}^2\right] \subset \Omega_\mathrm{c}. \end{aligned} \tag{5}$$

Where F^i_{U} and F^i_{L} are respectively equivalent to R^i_{U} and R^i_{L} in Equation (3). The solution of the problem in Equation (5) is the biggest rectangle one can fit in the feasible subspace, represented in Fig. 1 by the rectangle inside the white area: $F^1_U = 355$ kN, $F^1_L = 166$ kN, $F^2_U = 450$ kN, and $F^2_L = 355$ kN. All combinations of F^1 and F^2 inside these ranges are feasible and independent: the performance of component 1 positively contributes to the crashworthiness, independently of the performance of component 2, as well as the performance of component 2 in relation to component 1.

2.3 The need for more flexibility

Increasing the flexibility of the solution space method has been the objective of several works in the past. This search was performed both regarding the direct and the indirect method. On one hand, the research regarding the direct method focused on how to represent the feasible shape [5]. On the other hand, the research in the field of the indirect method for solution space looked into how to provide wider corridors or how to change them.

The increase of flexibility of the indirect method for evaluating the solution space focused on changing the independent set of corridors to either expand them or tailor them to certain development necessities. Both these approaches introduce some dependencies between the corridors. For example, in [4, 17], a temporal dependency is introduced between the components: they define an order to design the components. This time-based dependency allows them to project the feasible subspace on the smaller subspace defined by the variables of each component. By computing the corridors in the projected subspace, they find wider ranges per component. The second approach allows the developers to change any corridor by moving their limits at will [15]. This approach changes the surrounding sections to locally satisfy the crashworthiness requirements.

In the first method, the dependencies are introduced on a temporal base. Therefore, even if the corridors are computed with the same criteria in the projected subspace, the possibility of developing the component concurrently is lost when defining the order in which they need to be designed. In other words, the first component is developed within a large feasible space. The second one, on the contrary, is conceived according to the narrower corridors that depend on the design of the first component. The second method disregards the step of fitting an orthogonal base in the feasible subspace Ω_c. Consequently, the corridors computed in this method are representing a feasible subspace, but not necessarily an independent one.

The work we are presenting here aims at maintaining the complete independence of the computed corridors. This, firstly, means that we do not want to increase the size of the corridors by introducing a temporal dependency. Secondly, when tailoring the corridors to the need of the development, we want to provide new corridors that are guaranteed to be feasible and independent, so that the development of the components can continue concurrently. To avoid both these drawbacks, the computation is carried out on the whole feasible subspace, like in the solution space formulation of [3, 8] and not a projection of it. Moreover, when tailoring to a development need, the re-computed corridors re-impose also the independence condition.

Under a different perspective, the procedure we propose here allows a user to investigate an area of the feasible space without changing the vehicle configuration used to compute the solution space. This allows to investigate alternative solutions in the feasible space, without compromising any of the conditions imposed in the original formulation of the solution space method.

3 Adaptive solution space

We just saw how the increase of flexibility in the solution space methodology always meant to compromise on the independence of different components. As previously stated, we want to avoid such compromise to allow the same flexibility and freedom throughout the entire development. We propose here the adaptive solution space methodology to be able to tailor the corridors to the need of development while maintaining the feasibility and independence properties. We consider the needs of the development to be represented by an input defined by the developers after one or more iterations of the design of a component have been finalized. Therefore, we consider this methodology to be based on the solution space already computed, as, for example, the one presented in Sect. 2.1. The set already computed needs to be modified to account for the input given by the developers.

Suppose, now, that a component (component k) cannot meet the required performance over one or more sections, like the one presented later in Fig. 9. To re-compute the corridors, we first need to understand whether or not the performance measured on the component itself is within the feasible region Ω_c. Thus, the first step of this new methodology shows to the developers the feasible region related to component k. To do so, we project the corridors of all components except those of component k on the subspace described by this very last. Let us call the projected space $\Omega_{\text{proj-k}}$. To define $\Omega_{\text{proj-k}}$ consider the complement of Ω_c, i.e. the non-feasible subspace Ω_f:

$$\Omega_f = \overline{\Omega_c}. \tag{6}$$

$\Omega_{\text{proj-k}}$ is then defined as the complement of projection on the subspace of component k of the non-feasible region:

$$\Omega_{\text{proj-k}} = \overline{\Omega_{\text{f-k}}}. \tag{7}$$

Where $\Omega_{\text{f-k}}$ belongs to the space of $\boldsymbol{R}^k$, and is defined as:

$$\Omega_{\text{f-k}} = \left\{\boldsymbol{R} : \boldsymbol{R}^{k,\sim k} \in \Omega_f \ \forall \ \boldsymbol{R}^{\sim k}\right\}. \tag{8}$$

Where $\boldsymbol{R}^k$ are the feasible ranges assigned to component k, and $\boldsymbol{R}^{\sim k}$ those assigned to all other components, and $\boldsymbol{R}^{k,\sim k}$ is the set composed of both sets.

The feasible region of component k is computed by fitting an outer hypercube to $\Omega_{\text{proj-k}}$. Notice that, since we are fitting a hypercube to an unknown convex shape, it can contain also infeasible regions. In this case, the algorithm will fail to find new corridors and the designer will be warned.

The next step collects the inputs of a developer and re-computes the corridors to find the new set of force ranges. The developer-defined inputs can be expressed as a set of equality conditions to modify $\boldsymbol{R}$. We change the way the intervals can be positioned. The user-input can only fix the width, the central value, the lower limit, or the upper limit, of the corridors at a time. The constraints take the form of Equations (9)-(12):

$$R^i_{\text{width}} = R^i_U - R^i_L = w, \tag{9}$$

$$R^i_{\text{middle}} = \frac{R^i_U + R^i_L}{2} = m, \tag{10}$$

$$R^i_L = l, \tag{11}$$

$$R^i_U = u. \tag{12}$$

Where w, m, l, and u represent the user input as a constant value. We define $\boldsymbol{R}_{\text{c-user}}$ the set of conditions made of $\boldsymbol{R}$ and the extra conditions derived from the equalities defined by the user. This new set yields a subspace $\Omega_{\text{c-user}}$ in the feasible space equal to or smaller than Ω_{c}. Mathematically, we define it as an extension of Equation (1):

$$\Omega_{\text{c}} = \left\{\boldsymbol{F} \in \Omega_{\text{ds}} \,|\, f_{\text{perf}}(\boldsymbol{F}) \leq \boldsymbol{R}_{\text{c-user}}\right\} \subseteq \Omega_{\text{ds}}. \tag{13}$$

Where $\boldsymbol{F}$, Ω_{ds}, and f_{perf} have the same meaning as given in Sect. 2.1.

Under these new conditions, we reformulate the optimization problem to fit the biggest hypercube inside $\Omega_{\text{c-user}}$. The new optimization problem is expressed in Equation (14):

$$\begin{aligned} \max \quad & \prod_{i=1}^{\text{n}} \left[R^i_{\text{L}} - R^i_{\text{U}}\right] \\ \text{s.t.} \quad & \left[R^1_{\text{L}}, R^1_{\text{U}}\right] \times \left[R^2_{\text{L}}, R^2_{\text{U}}\right] \times \cdots \times \left[R^n_{\text{L}}, R^n_{\text{U}}\right] \subset \Omega_{\text{c-user}}. \end{aligned} \tag{14}$$

To solve this problem we use the interior point methods available in [16]. To do so, we reformulate the problem of Equation (13) in the form:

$$\boldsymbol{b_l} \leq \boldsymbol{Ax} \leq \boldsymbol{b_u}. \tag{15}$$

Where $\boldsymbol{b_l}$, and $\boldsymbol{b_u}$ are vectors containing all conditions defined in $\boldsymbol{R}_{\text{c-user}}$. $\boldsymbol{x}$ is a vector containing all the variables of the problem (i.e. R^i_{L}, R^i_{U}), and $\boldsymbol{A}$ is matrix containing the coefficients of the inequalities and equalities.

This formulation allows to find efficiently the maximum of Equation (14). In this way, we find a new set of corridors (i.e. a set of R^i_{L}, R^i_{U}) that fulfils the feasibility and independence requirements. The first is guaranteed by computing the solution space in a subset of the feasible space Ω_c. The second is guaranteed by solving the problem in Equation (14). Notice that the problem in this equation involves all the variables describing the solution space, thus affecting all corridors. On top of this, the conditions given by the developers allow the method to tailor the new corridors to the problems presented by the development itself. The re-computed corridors are a global adaptation of the original ones: they represent a different part of the feasible space. This space is equal or smaller than the one computed in Sect. 2.1. However, it allows the development to continue designing all components concurrently, have a crashworthy design at the end, and tailor the guideline provided by the solution space to the specific need of the development.

Although the procedure is very similar to the one originally proposed in [8], the one we propose is additionally capable of limiting the search of the solution to Equation (14) in an adaptive manner to a certain area of the feasible space. If we consider the problem of Sect. 2.1, instead of looking for the global maximum of Equation (3), we propose here a

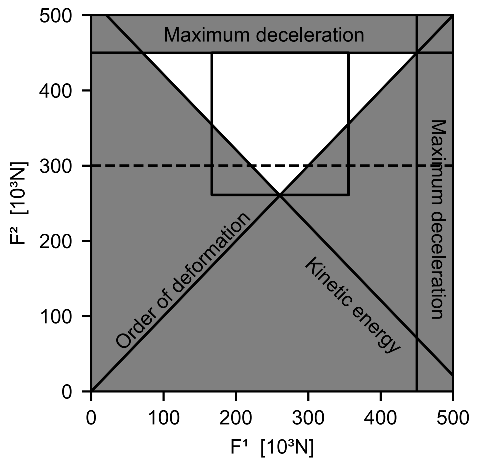

Figure 4 Representation of the two-dimensional design space and the limits of $\Omega_{\text{proj-k}}$ for component 2: the rectangle represents the subspace obtained by projecting component 1 corridors on the subspace described by the sections of component 2. The dashed line corresponding to F_2 = 300 kN represents the desired design points the developers want to include in the solution space

method that looks for a local maximum in a predetermined area. The previous formulation was not capable of this wished-for computation. It is emphasised here that the conditions of Equations (9)-(12) do not affect the feasible space. They are responsible only of diverting the search to a different area of it, which is an attractive feature of the method in later phases of development.

To show how the adaptive solution space method provides such new corridors, we provide in the coming sections two examples. The first applies the adaptive computation of solution space to the two-dimensional example already introduced. The second, instead, showcases how to use a force-deformation curve to adapt the corridors.

4 Academic example of computing adaptive solution space

With the first example, we showcase how the solution space is computed adaptively. Thus, we elaborate on the two-dimensional example introduced in Sect. 2.2. Suppose that the engineers have found a design for the second component at F^2 = 300 kN that better fits the vehicle structure. To include this design point, they want to move the lower limit of the range for the second component down at 280 kN. They can take advantage of the adaptive solution space methodology.

To modify the corridors representing the solution space, we follow the steps introduced in the previous section. In the first step, we project the corridor of component 1 on the subspace of component 2 to find the limits of the projected feasible space. $\Omega_{\text{proj-2}}$ is represented in Fig. 4 by the rectangle spanning across both the feasible and infeasible regions. Since the upper limit of $\Omega_{\text{proj-2}}$ is equal to 450 kN, and the lower limit is 261 kN, the condition on component 2 of bringing the lower limit to 280 kN is within the feasible subspace: it can be imposed. This user-defined constraint can be expressed by:

$$F^2_{\text{L}} = 280 \text{ kN}. \tag{16}$$

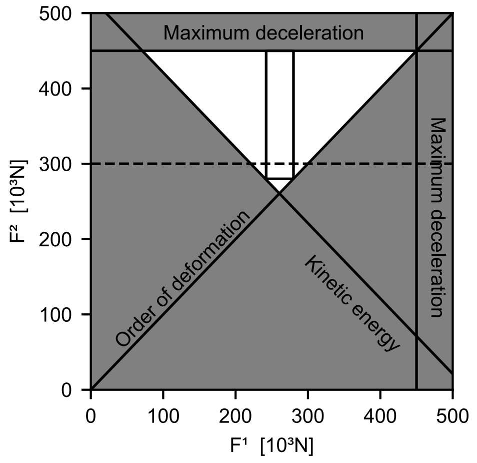

Figure 5 Representation of the two-dimensional design space and the adapted solution space: the new rectangle is computed under user-defined constraints. The dashed line corresponding to $F_2 = 300$ kN represents the desired design points the developers want to include in the solution space

It follows that the new feasible subspace $\Omega_{\text{c-user}}$, defined in Equation (13), yields to:

$$\Omega_{\text{c-user}}: \begin{cases} F^1 \leq 450 \text{ kN}; & \text{(maximum deceleration)}, \\ F^2 \leq 450 \text{ kN}; & \text{(maximum deceleration)}, \\ 0.23F^1 + 0.23F^2 \leq -121 \text{ kN}; & \text{(initial impact velocity)}, \\ F^1 - F^2 \leq 0; & \text{(order of deformation)}, \\ F^2_{\text{L}} = 280 \text{ kN}; & \text{(user-defined constraint)}. \end{cases} \tag{17}$$

And the optimization problem:

$$\begin{aligned} \max \quad & \prod_{i=1}^{2} \left[F^i_{\text{L}} - F^i_{\text{U}}\right] \\ \text{s.t.} \quad & \left[F^1_{\text{L}}, F^1_{\text{U}}\right] \times \left[F^2_{\text{L}}, F^2_{\text{U}}\right] \subset \Omega_{\text{c-user}}. \end{aligned} \tag{18}$$

By solving the problem in Equation (18) we find the new force ranges: $F^1_U = 280$ kN, $F^1_L = 242$ kN, $F^2_U = 450$ kN, and $F^2_L = 280$ kN. These are represented in Fig. 5 by the narrow rectangle. Notice that this new rectangle, although smaller, now includes the design point of component 2 of 300 kN. It also provides a new set of independent and feasible solution, thanks to the Cartesian product included in the optimization problem.

5 Practical application of adaptive solution space

The second example showcases how to use the information gained during development. Since we want to focus only on how to apply the adaptive solution space methodology, we, once again, borrow an example from the literature and build on top of it. In [2] a study case of Component Solution Space for the Honda Accord model [13] is presented. The example is always applied to the full rigid barrier test, hence the same performance functions presented for the 2d example in Sect. 2.2 apply also to this case, with due modifications. The vehicle is made of 7 components, highlighted in Fig. 6, which are divided into 31 sections.

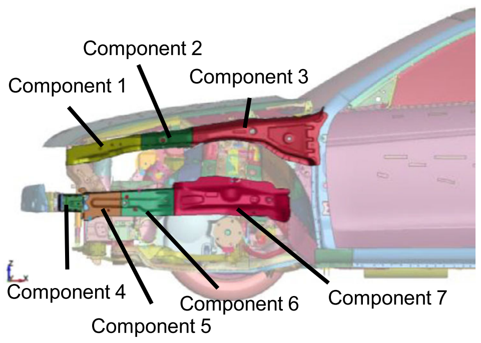

Figure 6 The 7 components of the Honda Accord [13] considered to build the analytical model for computing the solution space

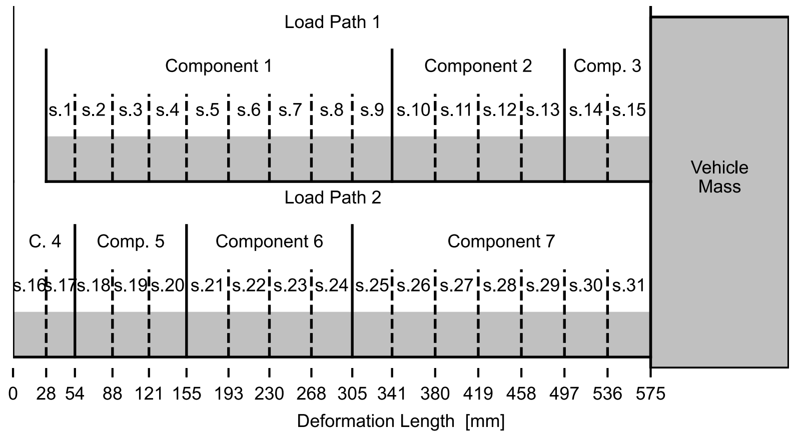

Figure 7 Deformation Space Model with 31 sections for the Honda Accord, constructed following the example from [2] and using the information from the Honda Accord model [13]

The Deformation Space Model is visible in Fig. 7, and, as shown in the figure, it serves to the analytical model for computing the solution space a total of 31 variables, corresponding to the 31 sections. The corridors of Component Solution Space for this second model of the Honda Accord are shown in Fig. 8. In it, the force ranges are represented against the deformation length fixed in the Deformation Space Model. For further details on the computation of the corridors we invite to refer to [2].

Besides a solution space, to adapt the corridors, we need to have some development data. To gain these, we optimize component number 7 in Fig. 6 for reducing its mass and being crashworthy [1]. In this work, the corridors from solution space act as constraints of the optimization and the objective is a function of the energy dissipated and of the mass of the component. From the optimized component we can measure the force-deformation curve, over-imposed on the sections corresponding to component 7 in Fig. 9. This curve represents the data to adapt the solution space: the force-deformation curve is split into the sections of component 7 and, over each section, we compute the average of the measured force. If in a section the mean value is outside the corridor, we adapt the section.

As highlighted by the blue lines in Fig. 9, sections 25, 26, 27, and 28 are outside the corridors of Component Solution Space. To adapt these, we first need to check if the measured

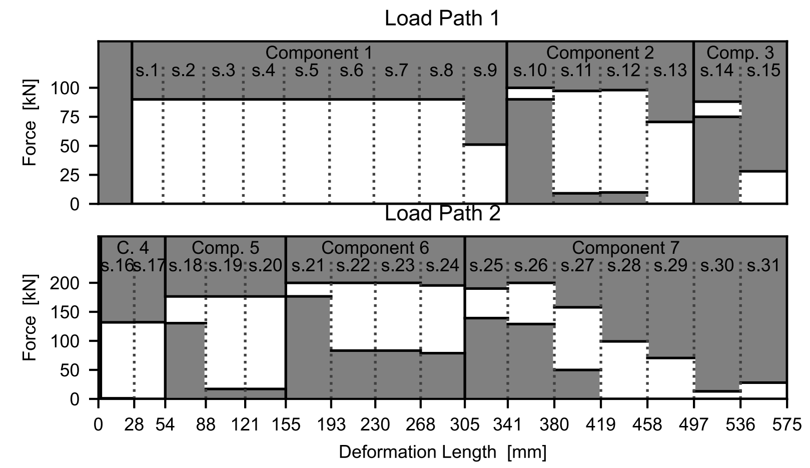

Figure 8 The corridors from solution space computed following the example from [2]

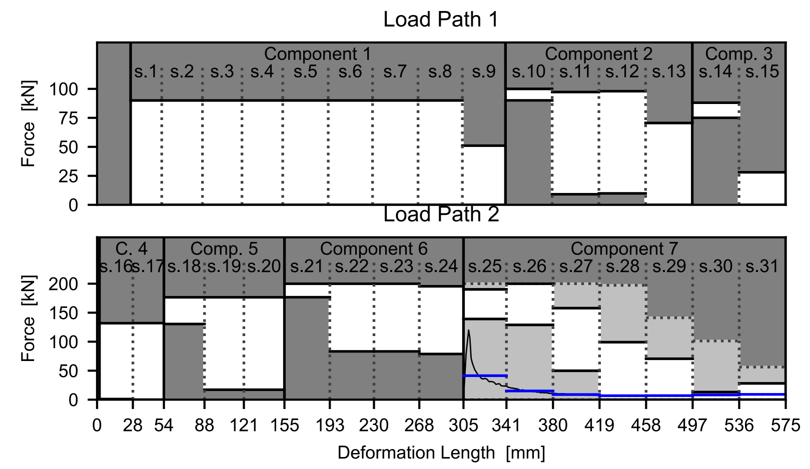

Figure 9 Feasible Space limits in light grey and force-deformation of component 7; in solid blue the average force in each section measured for component 7

values are inside the feasible space: $\Omega_{\text{proj}-7}$ for component 7 can be seen in Fig. 9 in light grey. The mean values of the force-deformation curve are inside $\Omega_{\text{proj}-7}$ as the figure itself shows. We can now define the conditions to adapt the corridors to the measured force-deformation curve. We impose that the middle value of the corridors must be equal to the measured value. Therefore, the user-defined constraints are expressed by the following set of equations:

$$\begin{cases} F^{25}_{\text{middle}} = 41\ \text{N}, \\ F^{26}_{\text{middle}} = 15\ \text{N}, \\ F^{27}_{\text{middle}} = 9\ \text{N}, \\ F^{28}_{\text{middle}} = 7\ \text{N}. \end{cases} \tag{19}$$

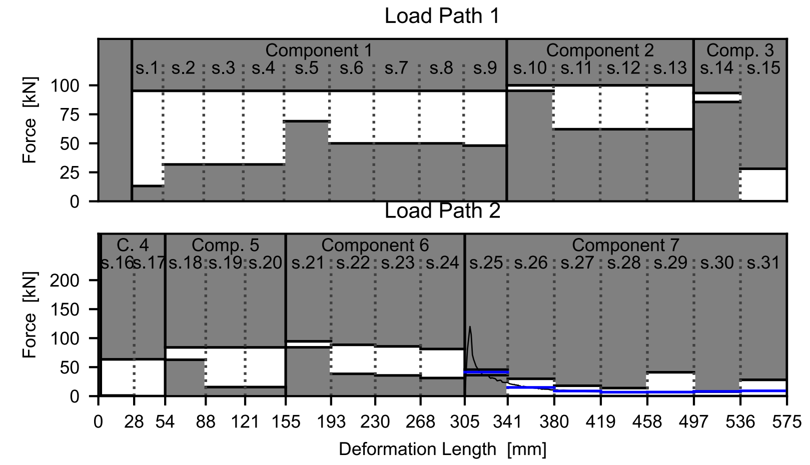

Figure 10 The solution space adapted to the performance of component 7

Where F^{25}_{middle}, F^{26}_{middle}, F^{27}_{middle}, and F^{28}_{middle} are the middle value of the force range of, respectively, sections 25, 26, 27, and 28. These conditions, along with those defined for computing the Component Solution Space, define $\Omega_{\text{c-user}}$. Thus, we can formulate and solve the optimization problem of Equation (14). The solution is represented in Fig. 10: the corridors of the changed sections have shifted downwards to match the curve measured from the optimized component 7.

The new corridors provide a new set of feasible solutions that are still all independent of each other. This is guaranteed by the problem formulation presented in Sect. 3. Although the solution includes the data from development, we notice a substantial reduction in corridor width over all sections. While some components still have a good amount of freedom for the developers to be flexible with their design, others, like components 2, 3, and 6, have much narrower corridors.

6 Discussion

The methodology here presented fulfils our goal: adapt the solution space and maintain both feasibility and independence between components. Engineers can now iteratively find a solution space that better fits the problems they face during development. On top of this, unlike previous works [4, 15, 17], the engineers can continue to develop their components concurrently, thanks to the ability of the new approach to maintain the independence property of the solution space.

However, this type of adaptation comes at a cost: the total volume of the solution space is smaller. All corridors are narrower and the solution for other components can be more demanding. For example, section 10 in Fig. 10 has a corridor so narrow that matching its request would demand an excessive development effort. This problem is overcome by the fact that, once the data is available, the method is cheap and quick to run. Therefore the solution space can be adapted several times to find another compromise. One that does not reduce excessively one or more sections. Engineers can now quickly and inexpensively iterate between different solution spaces to find the one that has the best trade-off between the different challenges they are facing. How to perform this iteration is, however, outside the scope of this paper.

As last remark, reducing the volume of the corridors also reduces the flexibility our designer have to design a component. This would be true if the corridors found could no longer be edited. The possibility to adapt the corridors several time allows to find a compromise. The adaptation reduces the flexibility of individual components, but it grants more design flexibility overall by being able to manipulate corridors within the feasible space. Therefore, the engineers can overcome a problem like the one presented in Sect. 5.

7 Conclusion

In this paper, we first looked at the solution space methodology along with existing methods to compute and adapt it. However, since these adaptive methods relax the initial conditions, we propose here a new and flexible way of computing the solution space. We can change the solution space according to the information gained during the first phase of development: we call this method adaptive solution space.

As the presented examples show, the adaptive solution space methodology can recompute the solution space under the additional user-defined conditions. Therefore, if the corridors originally provided by the solution space cannot be met, the user can, now, change them and find new independent and feasible sets that better fit the development process. The adaptive computation of the solution space allows to iteratively find the corridors that better fit the vehicle structure at hand. It allows to design all components for crashworthiness independently throughout the entire development, and it provides developers with the needed flexibility to find at a reduced effort an answer to the challenges coming from other disciplines of vehicle design.

Acknowledgements
We thank the two anonymous reviewers for their precious suggestions.

Funding
Open Access funding enabled and organized by Projekt DEAL. 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.

Availability of data and materials
Data will be made available on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author contributions
PA: conceptualization; PA, VAL: methodology; PA: software; PA: formal analysis and investigation; PA: writing—original draft preparation; PA, VAL, FD: writing—review and editing; VAL, FD: supervision. All authors read and approved the final manuscript.

Author details
[1]TUM School of Engineering and Design, Technical University of Munich, Arcisstr. 21, 80333, Munich, Germany. [2]Research and Innovation Center, BMW Group, Knorrstr. 147, 80788, Munich, Germany.

Received: 10 February 2023 Accepted: 19 September 2023 Published online: 12 October 2023

References
1. Burmberger L. Efficient Global Optimization of Structural Components for Solution Spaces in Vehicle Crash Design. M.Sc. Thesis, Technical University of Munich, Germany; 2020.
2. Daub M. Optimizing Flexibility for Component Design in Systems Engineering under Epistemic Uncertainty. PhD thesis. Technical University of Munich, Germany; 2020.
3. Daub M, Duddeck F, Zimmermann M. Optimizing component solution spaces for systems design. Struct Multidiscip Optim. 2020;61:2097–109. https://doi.org/10.1007/s00158-019-02456-8.

4. Daub M, Wöhr F, Zimmermann M. Optimizing distributed design processes for flexibility and cost. In: Proceedings of the 22nd international DSM conference (DSM 2020). 2020. p. 1–10.
5. Erschen S. Optimal decomposition of high-dimensional solution spaces for chassis design. PhD thesis. Technical University of Munich, Germany; 2018.
6. European New Car Assessment Programme, Full Width Frontal Impact Testing Protocol, Version 1.2.1, November 2021, URL: https://cdn.euroncap.com/media/67284/euro-ncap-frontal-fw-test-protocol-v121.pdf (visited on 11/08/2022)
7. Fender J. Solution Spaces for Vehicle Crash Design. PhD thesis. Technical University of Munich, Germany; 2013.
8. Fender J, Duddeck F, Zimmermann M. Direct computation of solution spaces. Struct Multidiscip Optim. 2017;55:1787–96. https://doi.org/10.1007/s00158-016-1615-y.
9. Graff L, Harbrecht H, Zimmermann M. On the computation of solution spaces in high dimensions. Struct Multidiscip Optim. 2016;54:811–29. https://doi.org/10.1007/s00158-016-1454-x.
10. Kim HM, Michelena NF, Papalambros PY, Jiang T. Target cascading in optimal system design. J Mech Des. 2003;125(3):474–80. https://doi.org/10.1115/1.1582501.
11. Lange VA, Fender J, Duddeck F. Relaxing high-dimensional constraints in the direct solution space method for early phase development. Optim Eng. 2018;19(4):887–915.
12. Lange VA, Fender J, Song L, Duddeck F. Early phase modeling of frontal impacts for crashworthiness: from lumped mass–spring models to deformation space models. Proc Inst Mech Eng, Part D, J Automob Eng. 2019;233(12):3000–15. https://doi.org/10.1177/0954407018814034.
13. National Highway Traffic Safety Administration, Crash Simulation Vehicle Models, URL: https://www.nhtsa.gov/crash-simulation-vehicle-models (visited on 12/06/2022).
14. National Highway Traffic Safety Administration, Laboratory Test Procedure for New Car Assessment Program Full Frontal Rigid Barrier Impact Testing 2018-05, URL: https://www.regulations.gov/document/NHTSA-2015-0046-0014 (visited on 11/08/2022).
15. Song L. Direkte Methoden zur Berechnung von Lösungsräumen in der Fahrzeugstrukturauslegung für Crash. M.Sc. Thesis, Technical University of Munich, Germany; 2013.
16. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17(3):261–72.
17. Vogt M, Duddeck F, Wahle M, Zimmermann M. Optimizing tolerance to uncertainty in systems design with early- and late-decision variables. IMA J Manag Math. 2019;30(3):269–80. https://doi.org/10.1093/imaman/dpy003.
18. Zimmermann M, Edler von Hoessle E. Computing solution spaces for robust design. Int J Numer Methods Eng. 2013;94:290–307. https://doi.org/10.1002/nme.4450.

Publisher's Note