OPTIMIZING REQUIREMENTS FOR MAXIMUM DESIGN FREEDOM CONSIDERING PHYSICAL FEASIBILITY

Rodrigues Della Noce, Eduardo;
Zimmermann, Markus

Technical University of Munich (TUM)

ABSTRACT

Solution spaces are sets of designs that meet all quantitative requirements of a given design problem, aiding requirement management. In previous works, ways of calculating subsets of the complete solution space as hyper-boxes, corresponding to a collection of permissible intervals for design variables, were developed. These intervals can be used to formulate independent component requirements with built-in tolerance. However, these works did not take physical feasibility into account, which has two disadvantages: first, solution spaces may be useless, when the included designs cannot be realized. Second, bad designs that are not physically feasible unnecessarily restrict the design space that can be used for requirement formulation.

In this paper, we present the new concept of a requirement space that is defined as the largest set of designs that (1) allows for decomposition (e.g., into intervals when it is box-shaped), (2) maximizes the useful design space (good and physically feasible), and (3) excludes the non-acceptable design space (bad and physically feasible). A small example from robot design illustrates that requirement spaces can be significantly larger than solution spaces and thus improve requirement decomposition.

Keywords: Requirements, Complexity, Concurrent Engineering (CE), Solution Spaces, Decomposition

Contact:
Rodrigues Della Noce, Eduardo
Technical University of Munich (TUM)
Germany
eduardo.noce@tum.de

Cite this article: Rodrigues Della Noce, E., Zimmermann, M. (2023) ‘Optimizing Requirements for Maximum Design Freedom Considering Physical Feasibility’, in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.287

1 INTRODUCTION

Complexity in the context of systems design is strongly related to the involved uncertainties in the development process. When a system includes many design variables and teams, decisions must be made with incomplete information, which can make them difficult (Suh, 2005; Parkinson et al., 1993). Reasons for this include manufacturing problems, detailed design modifications in later stages of the process, and changes in operation conditions (Daub et al., 2020). Under these circumstances, classical optimization approaches with an associated performance function which is minimized (as shown in Avriel et al. (1973); Boyd et al. (2004)) may have their performance deteriorated by such uncertainties, and may even violate requirements. New levels of complexity are added when also considering many design objectives (which may have non-linear dependencies with the design variables) and the development of product families (where different products may share components between them) (Jiao et al., 2007; Eichstetter et al., 2015).

Methods for robust design have been developed over the years to deal with the increasing complexity. In Beyer and Sendhoff (2007), many different approaches to dealing with uncertainties are presented, using what is called robust design optimization (RDO). RDO leads to a design that combines an optimum of system performance and its variation due to a prescribed change in the design and environmental variables. The issue with this approach is that it requires said change (normally via a probabilistic distribution), and that is often also not known beforehand. Much of the same can be said for other quantitative uncertainty propagation methods (Le Maître and Knio, 2010), such as reliability-based design optimization (RBDO) (Zhao and Ono, 1999; Gunawan and Papalambros, 2006; Mourelatos and Liang, 2005).

A different type of strategy revolves around not trying to find one particular design, but instead finding an appropriate group of designs; this is called set-based design (Al-Ashaab et al., 2009; Sobek et al., 1999). Comprehensive studies on the current state and development of set-based design strategies have been performed (Shallcross et al., 2020; Dullen et al., 2021), showing the growing interest in the topic. These approaches hold the advantage that they are able to give more freedom to the design teams while allowing them to work concurrently. Of particular interest to this work is the set-based design that uses so-called solution spaces, which are sets of designs that meet all the system requirements.

Solution spaces were initially conceptualized in Lehar and Zimmermann (2012) and more formally introduced in Zimmermann and von Hoessle (2013). This approach separates the input space into a good region (where all requirements are met, composed therefore of the appropriately named good designs) and a bad region (where at least one requirement is violated, composed of so-called bad designs). Moreover, the introduced method would compute intervals of permissible designs for each design variable, allowing for independent work on each one with some design freedom. This allowed for concurrent engineering in the development process, while also accounting for uncertainties without requiring an uncertainty model.

One pitfall of the current solution space-based methods is that, so far, physical feasibility of the designs has not been taken into consideration during the computation. In this work, a certain design being physically feasible means it has a correspondent detailed design, and so, can actually be realized. This has two disadvantages: first, the computed solution space may contain few or no physically feasible designs, which might nullify the advantages of the method; second, it may lead to over-restrictive intervals, since designs that are bad but physically infeasible (and so, cannot be realized, and will not lead to a system that cannot fulfill its requirements) are also excluded from the intervals.

In this work, therefore, we aim to introduce a new concept, called *requirement spaces*, where physical feasibility may be taken into account to obtain sets of designs which are more useful. The term "requirement spaces" was used before in Ponn and Lindemann (2011), but with a different meaning (there, it is used to express the space of development goals and constraints), and no other use of this term was found in the literature at the time of writing this work. Here, a "requirement space" is a set of designs that are to be pursued in order to produce a physically feasible solution.

This paper is organized as follows: in Section 2, the classical solution space formulation is revisited and explained, and some of its applications and expansions are described; in Section 3, an example is shown where such classical formulation can lead to a region where no designs can be realized, or to a relatively small space; Section 4 is where the concept of requirement spaces is introduced, along with a more general problem statement; in Section 5 an application example in the form of an industrial robot is described, showcasing the advantages of the new method; finally, in Section 6, a summary of this paper is given, as well as general conclusions and an outlook for future research.

2 STATE OF THE ART

2.1 Definitions

Assume there is a model of the system available, but the values of the design variables are not yet set. There are a total of d design variables, which are collectively represented by vector $\mathbf{x} \in \mathbb{R}^d$. Similarly, there are a total of m quantities of interest (performance metrics) for this system. Each quantity of interest has a map that relates it to the design variables, as per the system model; such output function is represented by $f_j(\mathbf{x}), j = 1, ..., m$. For every quantity of interest, there is also a critical value (requirement) which it must not exceed, named $y_{jc}, j = 1, ..., m$. The design space is indicated by Ω_{ds}. A design that meets all of the requirements is called a good design, and a design that violates at least one requirement is called a bad design. The considered measure of a space, which is generally its area/volume/hyper-volume, is represented by $\mu(\Omega)$. The physical feasibility indicator $p(\mathbf{x})$ is a function such that the output is less or equal to zero if the design is physically feasbile, or greater than zero if not. The complete solution space is designated by $\Omega_g = \{\mathbf{x} \mid f_j(\mathbf{x}) \leq y_{jc}, j = 1, ..., m\}$, the complete physically feasible space is designated by $\Omega_p = \{\mathbf{x} \mid p(\mathbf{x}) \leq 0\}$, and the intersection between the two is written as $\Omega_{gp} = \Omega_g \cap \Omega_p$. The complete space of physically infeasible designs is designated by $\Omega_q = \{\mathbf{x} \mid p(\mathbf{x}) > 0\}$.

2.2 Problem statement

The classical goal when calculating solution spaces is to find the set of designs that has the largest measure possible and that does not include any designs that do not meet the system requirements. Mathematically, that reads as follows:

$$\begin{aligned} &\max_{\Omega} \mu(\Omega) \\ &\text{subject to: } f_j(\mathbf{x}) \leq y_{jc} \ \forall \mathbf{x} \in \Omega, j = 1, ..., m \end{aligned} \tag{1}$$

If the sought solution is (hyper-)box-shaped, then the space Ω has the particular form $\Omega = \Omega(\xi) = [x_{1l}, x_{1u}] \times ... \times [x_{dl}, x_{du}]$, where $\xi = (x_{1l}, x_{2l}, ..., x_{(d-1)u}, x_{du})$ and x_{il}, x_{iu} are the lower and upper bounds respectively of the permissible intervals for the i-th design variable x. The hyper-volume of the solution space, in this case, is given by $\mu(\Omega) = \prod_{i=1}^{d}(x_{iu} - x_{il})$.

A sampling-based algorithm used to solve this problem was originally shown in Zimmermann and von Hoessle (2013) and was further expanded upon in Graff et al. (2016). An example of how the result can look is shown in Figure 1, where this method was applied to a two-dimensional example where the set of good designs forms a hollow circle.

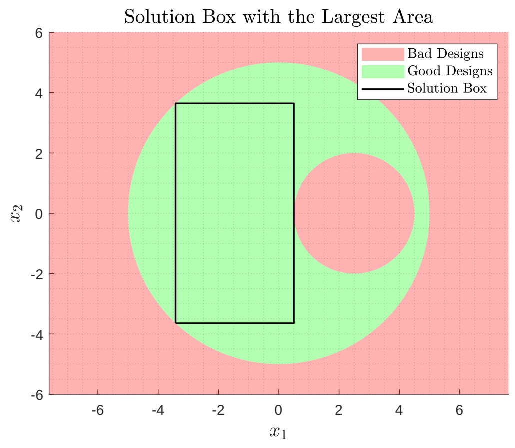


Figure 1. Solution box with largest area for this hollow circle example; the edges of the box represent the limits of the intervals for each design variable.

https://doi.org/10.1017/pds.2023.287 Published online by Cambridge University Press

2.3 Applications and extensions

The solution spaces approach was first developed and used in the context of vehicle crash design (Lehar and Zimmermann, 2012; Zimmermann and von Hoessle, 2013; Fender, 2013; Zimmermann et al., 2017) in order to deal with the high degree of uncertainty present in the development process. Since then, it has also seen use in the development of product families for electric vehicles (Eichstetter et al., 2015; Rötzer et al., 2020, 2022) and on the integration design of a powertrain (Stumpf et al., 2022).
As for the extensions of the classical method, many different approaches have been developed with the goal of increasing the size of the resulting solution space. Solution-compensation spaces, for example, separate the problem into early- and late-decision variables in order to give more design-freedom to the early-decision variables, while still ensuring there's a degree of built-in tolerance for the late-decision variables (Vogt et al., 2018; Stumpf et al., 2022). A different approach involves introducing pair-wise coupling of design variables, either through enabling rotation of the box or by introducing 2d-polygons for paired variables, which decreases the restrictiveness of the problem and thus increases the overall size of the solution space (Erschen et al., 2018; Harbrecht et al., 2019, 2021). Another approach involves coupling any number of design variables, forming so-called component solution spaces, though at the time of writing, this is restricted to problems with linear output functions only (Daub et al., 2020).
The new theory shown in this work is framed in the context of the classical formulation of solution spaces, but could be adapted and used in any of the aforementioned extensions.

3 PROBLEM DESCRIPTION

Solution spaces are used in the systems design phase of the development process. It may be that not all designs that are part of the design space are physically feasible designs. In this context, a design being physically feasible means it has a correspondent detailed design, and as such, can be realized when proceeding to the component design phase. An example of a physically infeasible design would be a set of stiffness components for a structure that are too high for the given geometric constraints and material available (Krischer and Zimmermann, 2021). A different example, explored in Section 5, is motors with given combinations of torque and mass which simply are not available in the market and might not be able to be built with current technology. It is important to distinguish between physical feasibility and requirements: physical feasibility is viewed from a bottom-up perspective, being a property of a given design which cannot be altered, and does not depend on the wishes of a designer; on the other hand, a requirement is viewed from a top-down perspective, and expresses what a designer wants to achieve.
Current methods for the computation of solution spaces do not take into account whether a design is feasible or not, only if it meets the given system requirements. To show how this may lead to useless or overly-restrictive results, the two-dimensional example shown in Figure 2 will be used. This example has similar characteristics to the application problem shown in Section 5, but was crafted to further exacerbate the differences and improve the understanding of the concepts.

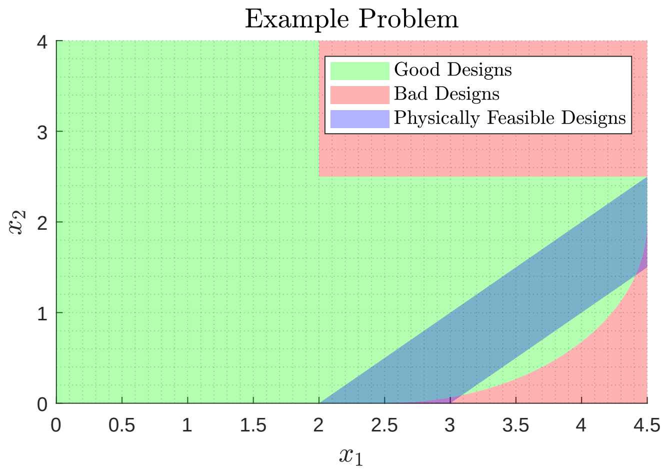
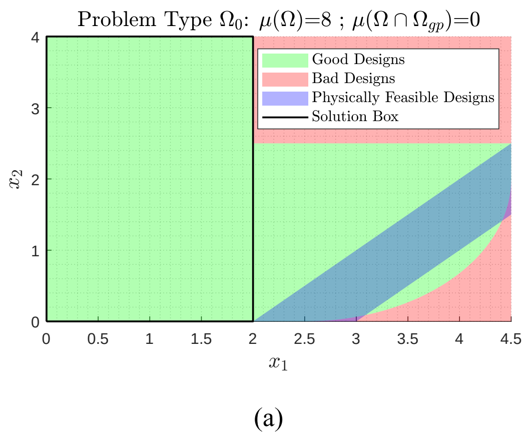


Figure 2. Example problem used in this section to illustrate the short-comings of the current formulation of the computation of solution spaces.

https://doi.org/10.1017/pds.2023.287 Published online by Cambridge University Press

A designer will be interested only in the designs that are simultaneously good and physically feasible, since those are the ones that meet all system requirements and can be realized. When computing the solution space using the formulation shown in Equation 1, however, no physically feasible design will be included in the final result, as shown in Figure 3a. This is due to the shape of the region of good and bad designs, which makes the left-hand side of the domain contain many more good designs then the right. The classical problem formulation will hereby be referred to as problem type Ω_0.
The initial way of dealing with this, without changing the problem statement, would be to introduce physical feasibility as a requirement. While that is possible, it leads to an overly-restrictive result, as shown in Figure 3b. Now all designs in the solution box are physically feasible, but many of the designs that both physically feasible and good are left out. This formulation will be referred to as problem type Ω_1.



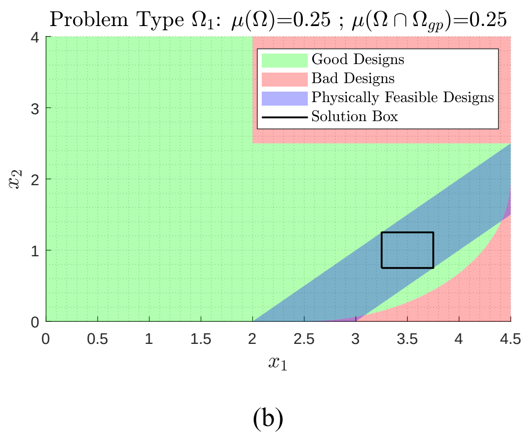


Figure 3. Largest solution box when using the classical approach for the example problem. In one case, the resulting box is large but contains no physically feasible design; on the other, all designs are physically feasible, but the result is overly restrictive.

Ideally, the solution box would focus around the points which are both good and physically feasible, but would still allow for physically infeasible designs to be a part of the solution if that meant including more of these useful designs. However, there's no way to express this in the original problem statement, showing there is an inherent limitation to it.

4 REQUIREMENT SPACES THEORY

As alluded to in the previous section, the current problem statement is limited in its form, since it cannot separate between designs that can be included in the solution box and designs whose presence in the solution box should be maximized. In order to deal with that, a new problem statement is introduced, which reads as follows:

$$\begin{aligned} &\max_{\Omega} \mu(\Omega \cap \Omega_{use}) \\ \text{subject to: } &\Omega \subseteq \Omega_{acc} \text{ (acceptability condition)} \\ &\forall (i, x_i \in \partial\Omega_i), \exists \mathbf{x} \in \Omega_{use} \text{ (leanness condition)} \end{aligned} \tag{2}$$

Two new spaces where introduced, Ω_{acc} and Ω_{use}. Ω_{acc} is the acceptable space, and is closely related to the acceptability condition; essentially, the acceptable space defines which designs will be allowed inside the box, and the acceptability condition is what substitutes the previous constraint that all designs in the space must meet all requirements. Ω_{use}, on the other hand, is the useful space; it defines which designs will actually contribute to the objective function when included in the space Ω. An important distinction is now relevant: up until now, Ω always only contained good designs, and so, it could be called a solution space. With this new problem statement, however, the accepted space does not need to be only composed of, or even include (on a theoretical level), good designs. As such, Ω is now denominated as requirement space. Note that a requirement space can also be a solution space if it only includes good designs, but that is not imposed anymore.

https://doi.org/10.1017/pds.2023.287 Published online by Cambridge University Press

There is another new component to the problem statement, and that is the leanness condition. The leanness condition is included because, without it, the problem may become ill-posed depending on the distribution of the useful designs. For example, assume a case where all designs are acceptable, but not all are useful; in said case, all boxes that include all useful designs would share the same value of the objective function between themselves, and the problem would not have a unique solution. Not only is that a problem mathematically, but it also doesn't represent the wanted behavior: designs that are not useful are being allowed to be inside the resulting requirement space only so the space of useful designs is larger; they are not desired by themselves, and should be eliminated if that wouldn't result in losing any useful space. The leanness condition accomplishes that; it is an additional constraint, which says that for every edge of the requirement space box, there must be one design that is useful and lies on that edge. This is illustrated in Figure 4.

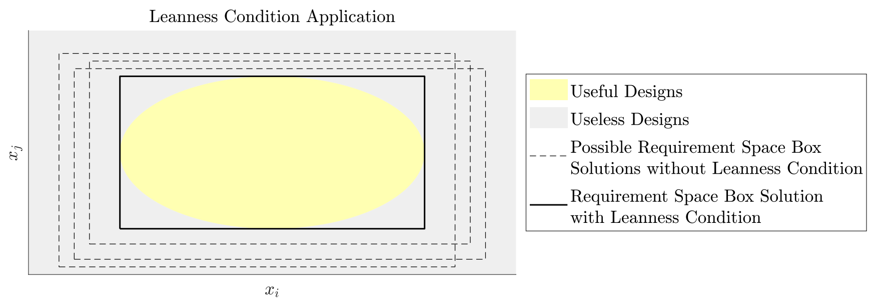

Leanness Condition Application
x_j
x_i
Useful Designs
Useless Designs
Possible Requirement Space Box Solutions without Leanness Condition
Requirement Space Box Solution with Leanness Condition

Figure 4. Application of the leanness condition, turning an ill-posed problem into a well-posed one, while not removing any useful designs.

Note, Ω_0 and Ω_1 can also be put into this framework, by considering $\Omega_{acc} = \Omega_g$ and $\Omega_{acc} = \Omega_{gp}$ respectively, as well as $\Omega_{use} = \Omega_{ds}$ for both. This shows the requirement space problem statement in 2 is a generalization of the solution space problem statement in 1.
Now, one can define new problem types by using the suggested framework. In order to allow for all good designs to be inside the solution results, while focusing on the physically feasible designs, one can consider $\Omega_{acc} = \Omega_g$ and $\Omega_{use} = \Omega_p$. This leads to problem type Ω_2, and the result is shown in Figure 5a.
It can be observed that now the requirement space is more suitable compared to the result with Ω_1, given that the area of designs that both and physically feasible has increased by a factor of approximately 3.2. In a real application, this would directly translate to more design freedom for the design teams. Note the bottom-right corner is restricted by the region of bad designs, while the bottom-left and top-right corners are restricted by the leanness condition, since expanding from where they currently are would not increase the amount of useful designs inside the resulting requirement space.
However, the requirement space can still be further improved. Assuming the physical feasibility indicator is globally reliable, one can consider relaxing the acceptability conditions further and allowing bad designs inside the box as well, as long as those are physically infeasible. If a design is physically infeasible, by definition, there will be no correspondent detailed design of it, and as such, a design team will not be able to build a component from it, and that means such a choice will not lead to a system that does not meet the requirements; simply another design will have to be chosen instead, with no further loss of time or money. By relaxing the problem as such, it is possible that more designs that are good and physically feasible might be included. This defines problem type Ω_3, where $\Omega_{acc} = \Omega_g \cup \Omega_q$ and $\Omega_{use} = \Omega_p$. The result of using this problem type for the example problem is shown in Figure 5b.
For this example problem, the solution box for Ω_3 contains a factor of 2.2 more good and physically feasible designs than Ω_2, or approximately 6 times more than Ω_1, which is a considerable improvement. It is important to keep in mind, however, that problem type Ω_3 depends heavily on a good physical feasibility indicator for both the existence and nonexistence of given designs, which can be challenging (especially for the nonexistence side, which is in fact the critical one for this problem type); if that is not the case, a design that does not meet the requirements and is actually physically feasible might end up being chosen, which could lead to problems later. In those cases, Ω_2 might be a better choice,

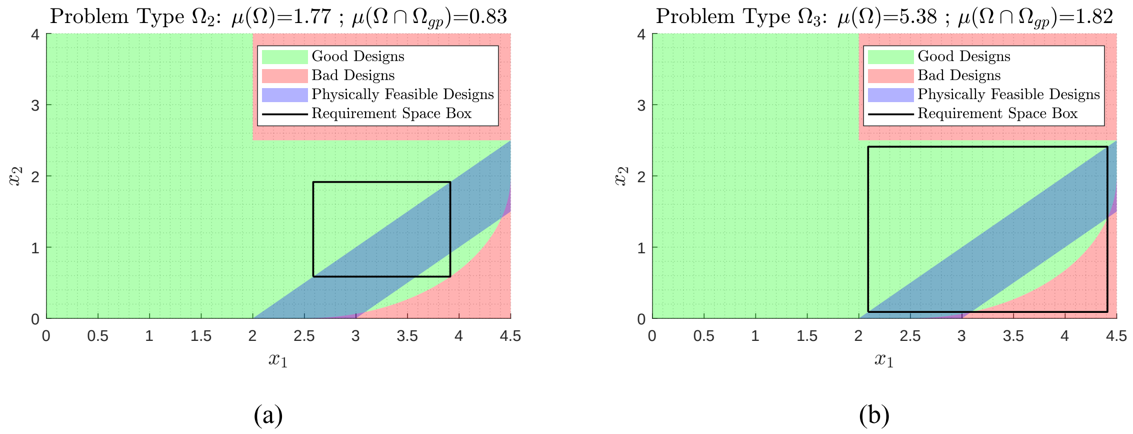


Figure 5. Largest solution box when using the new approach for the example problem. In one case, only good designs are accepted; on the other, bad designs are also accepted if they are physically infeasible.

as all designs inside it are good by definition. Table 1 summarizes the definitions and recommended applicability of the four problem types in accordance to the requirement space problem statement.

Table 1. Summary of the different problem types.

Type	Ω_{acc}	Ω_{use}	Application
Ω_0	Ω_g	Ω_{ds}	No information is known about the physical feasibility, or every design is physically feasible
Ω_1	Ω_{gp}	Ω_{ds}	-
Ω_2	Ω_g	Ω_p	Physical feasibility indicator is reliable for what exists, but not necessarily for what doesn't
Ω_3	$\Omega_g \cup \Omega_q$	Ω_p	Physical feasibility indicator is reliable for both existence and nonexistence of any design

5 APPLICATION EXAMPLE

5.1 Description

To illustrate the use of requirement spaces, an application with an industrial robot is considered. The robot has a two-link structure, with one motor at the base of each link, and the given task is to complete a given periodic movement. A simplified model is taken, where the dynamics are reduced to the 2D-plane and it is considered each link has to move sequentially, and not in parallel. The two motors are also assumed to be the same, in order to make the visualization of results easier. On the system level, there are imposed three requirements: one is on the cycle time required to complete the periodic movement, another is on the energy used during that movement, and finally, the last one is on the absolute velocity of the end effector. This choice of requirements is simple but fairly realistic: in any industrial application, it is absolutely desirable for the robot to complete its task and get ready for the next one as soon as possible, but simply brute-forcing the system with extremely high torques makes it also costs more during operation (from the electrical energy used); additionally, safety measures must be considered, especially if any sort of human interaction may occur. A sketch of the considered industrial robot is shown in Figure 6. For the designs variables, then, at least two are necessary: the operational torque of the motors, and their mass. To determine which motors are physically feasible, data from Padilla-Garcia et al. (2018) was used.

5.2 Results

An adapted algorithm based on the work presented in Zimmermann and von Hoessle (2013); Graff et al. (2016) was used to find the boxes with the biggest areas; in a nutshell, this algorithm considers a

https://doi.org/10.1017/pds.2023.287 Published online by Cambridge University Press

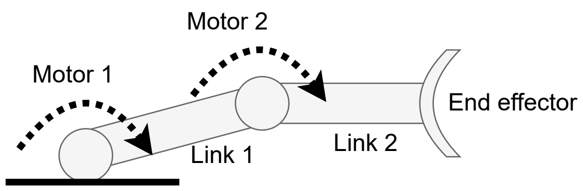


Figure 6. Sketch of the industrial robot considered in the application.

candidate box which iteratively expands and then gets trimmed based on samples which are evaluated inside it, with the goal of always finding the biggest possible box that only contains acceptable designs. For requirement spaces, an extra step at the end is added to apply the leanness condition. The results from each problem type are shown in Figure 7; as it can be observed, Ω_3 had the best results, with a resulting area of good and physically feasible designs that is approximately 85% larger than that of Ω_1 and approximately 15% larger than that of Ω_2.

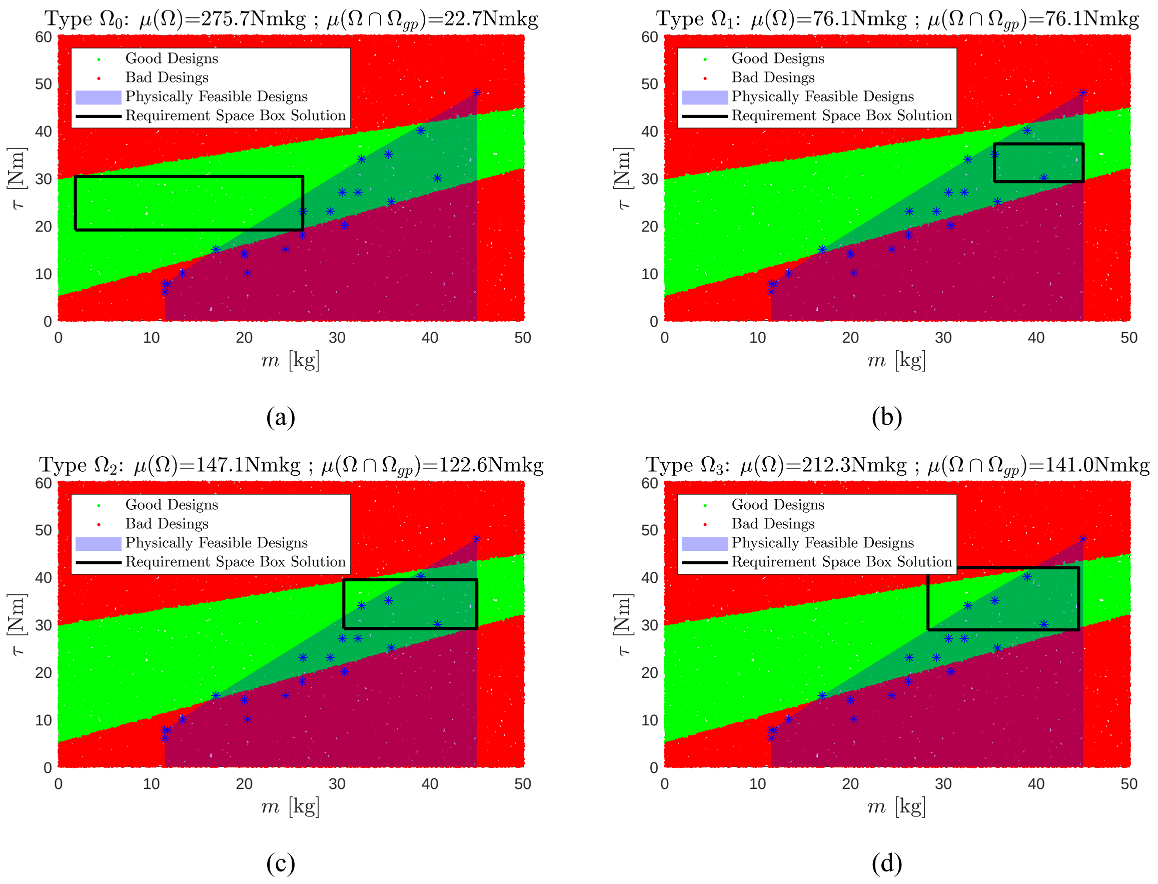


Figure 7. Solution and requirement space boxes for the industrial robot application problem.

5.3 Discussion

As can be observed, by using Ω_2, we were able to include more good and physically feasible designs than what was possible with the previous problem formulation, a result which was then further improved upon with Ω_3. Important to note, however, that this is a case where the physical feasibility indicator may not be so reliable; with the data, it is easy to show such motors exist, but there's no proof that others outside the delimited area do not, and as such, the result of Ω_3 may include some bad designs that can actually be realized. In such a case, Ω_2 is better suited for use.

In general, by using this method in the early design phase, a designer can generate better requirements for the components of the system, which can then be used in the later phases to make concurrent design

https://doi.org/10.1017/pds.2023.287 Published online by Cambridge University Press

decisions. In this example, with the new requirements in hand, a component designer could then proceed to look for motors commercially available which meet these requirements and make a decision based on a defined criterion, such as cost, without worrying if the chosen design meets the system level requirements, independent of what other component designers choose (as long as all of them follow the generated requirements).

6 SUMMARY AND CONCLUSION

In this paper, the concept of requirement spaces was introduced. A review of the classical method of solution space computation was performed, and its short-comings when it comes to considering physical feasibility were made clear with an example. After that, a new problem statement was introduced, which was shown to be a generalization of the previous one. With it, new problem types were defined, and it was shown that in the example they could lead to areas of good and physically feasible designs that were 2 to 6 times better than what was possible before. Finally, an application example with an industrial robot was also considered, where the new problem types showed results up to 85% better than those of classical types. Of note is that, much like solution spaces depend heavily on the accuracy of the system model to properly guide design choice, requirement spaces methods depend heavily on the accuracy of the physical feasibility model, which may be a limiting factor to the application of the method.
In the future, it would be important to extend the concepts shown here for the computation of boxes to more generalized versions of the solution spaces method, such as component solution spaces and solution-compensation spaces. By incorporating this, even more design freedom might be achieved.

ACKNOWLEDGMENTS

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 454149634.

REFERENCES

Al-Ashaab, A., Howell, S., Usowicz, K., Anta, P. and Gorka, A. (2009), "Set-based concurrent engineering model for automotive electronic/software systems development", *Competitive Design - Proceedings of the 19th CIRP Design Conference*.

Avriel, M., Wilde, D.J. and Rijckaert, M.J. (1973), *Optimization and design; International Summer School on the Impact of Optimization Theory on Technological Design*, Prentice-Hall Englewood Cliffs, N.J.

Beyer, H.G. and Sendhoff, B. (2007), "Robust optimization – a comprehensive survey", *Computer Methods in Applied Mechanics and Engineering*, Vol. 196 No. 33, pp. 3190–3218, http://doi.org/10.1016/j.cma.2007.03.003.

Boyd, S., Boyd, S., Vandenberghe, L. and Press, C.U. (2004), *Convex Optimization*, No. Teil 1 in Berichte über verteilte messysteme, Cambridge University Press.

Daub, M., Duddeck, F. and Zimmermann, M. (2020), "Optimizing component solution spaces for systems design", *Structural and Multidisciplinary Optimization*, Vol. 61, http://doi.org/10.1007/s00158-019-02456-8.

Dullen, S., Verma, D., Blackburn, M. and Whitcomb, C. (2021), "Survey on set-based design (sbd) quantitative methods", *Systems Engineering*, Vol. 24 No. 5, pp. 269–292, http://doi.org/10.1002/sys.21580.

Eichstetter, M., Müller, S. and Zimmermann, M. (2015), "Product family design with solution spaces", *Journal of Mechanical Design*, Vol. 137 No. 12, p. 121401, http://doi.org/10.1115/1.4031637.

Erschen, S., Duddeck, F., Gerdts, M. and Zimmermann, M. (2018), "On the optimal decomposition of high-dimensional solution spaces of complex systems", *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems*, Vol. 4 No. 2, pp. –, http://doi.org/10.1115/1.4037485. Solution space method, design optimization.

Fender, J. (2013), *Solution spaces for vehicle crash design*, Ph.D. thesis, Technische Universität München, Konstanz. PhD thesis.

Graff, L., Harbrecht, H. and Zimmermann, M. (2016), "On the computation of solution spaces in high dimensions", *Structural and Multidisciplinary Optimization*, Vol. 54 No. 4, pp. 811–829, http://doi.org/10.1007/s00158-016-1454-x.

Gunawan, S. and Papalambros, P.Y. (2006), "A Bayesian Approach to Reliability-Based Optimization With Incomplete Information", *Journal of Mechanical Design*, Vol. 128 No. 4, pp. 909–918, http://doi.org/10.1115/1.2204969.

Harbrecht, H., Tröndle, D. and Zimmermann, M. (2019), “A sampling-based optimization algorithm for solution spaces with pair-wise-coupled design variables”, *Structural and Multidisciplinary Optimization*, Vol. 60 No. 2, pp. 501–512, http://doi.org/10.1007/s00158-019-02221-x.

Harbrecht, H., Tröndle, D. and Zimmermann, M. (2021), “Approximating solution spaces as a product of polygons”, *Structural and Multidisciplinary Optimization*, http://doi.org/10.1007/s00158-021-02979-z.

Jiao, J., Simpson, T. and Siddique, Z. (2007), “Product family design and platform-based product development”, *Journal of Intelligent Manufacturing*, Vol. 18, pp. 1–3, http://doi.org/10.1007/s10845-007-0001-4.

Krischer, L. and Zimmermann, M. (2021), “Decomposition and optimization of linear structures using meta models”, *Structural and Multidisciplinary Optimization*, Vol. 64, http://doi.org/10.1007/s00158-021-02993-1.

Le Maître, O.P. and Knio, O.M. (2010), *Spectral Methods for Uncertainty Quantification*, Scientific Computation, http://doi.org/10.1007/978-90-481-3520-2.

Lehar, M. and Zimmermann, M. (2012), “An inexpensive estimate of failure probability for high-dimensional systems with uncertainty”, *Structural Safety*, Vol. 36-37, pp. 32–38, http://doi.org/10.1016/j.strusafe.2011.10.001.

Mourelatos, Z.P. and Liang, J. (2005), “A Methodology for Trading-Off Performance and Robustness Under Uncertainty”, *Journal of Mechanical Design*, Vol. 128 No. 4, pp. 856–863, http://doi.org/10.1115/1.2202883.

Padilla-Garcia, E.A., Rodriguez-Angeles, A., ReséNdiz, J.R. and Cruz-Villar, C.A. (2018), “Concurrent optimization for selection and control of ac servomotors on the powertrain of industrial robots”, *IEEE Access*, Vol. 6, pp. 27923–27938, http://doi.org/10.1109/ACCESS.2018.2840537.

Parkinson, A., Sorensen, C. and Pourhassan, N. (1993), “A General Approach for Robust Optimal Design”, *Journal of Mechanical Design*, Vol. 115 No. 1, pp. 74–80, http://doi.org/10.1115/1.2919328.

Ponn, J. and Lindemann, U. (2011), *Konzeptentwicklung und Gestaltung technischer Produkte: Systematisch von Anforderungen zu Konzepten und Gestaltlösungen*, VDI-Buch, Springer Berlin Heidelberg.

Rötzer, S., Berger, V. and Zimmermann, M. (2022), “Cost optimization of product families using solution spaces: Application to early-stage electric vehicle design”, in: P. of the Design Society (Editor), *Design 2022*, Cambridge University Press (CUP).

Rötzer, S., Thoma, D. and Zimmermann, M. (2020), “Cost optimization of product families using solution spaces”, in: *Proceedings of the Design Society: DESIGN Conference*, Cambridge University Press (CUP), pp. 1087–1094, http://doi.org/10.1017/dsd.2020.178.

Shallcross, N., Parnell, G.S., Pohl, E. and Specking, E. (2020), “Set-based design: The state-of-practice and research opportunities”, *Systems Engineering*, Vol. 23 No. 5, pp. 557–578, http://doi.org/10.1002/sys.21549.

Sobek, D., Ward, A. and Liker, J. (1999), “Toyota’s principles of set-based concurrent engineering”, *Sloan Management Review*, Vol. 40.

Stumpf, J., Cóndor López, J.G., Naumann, T. and Zimmermann, M. (2022), “Systems design using solution-compensation spaces with built-in tolerance applied to powertrain integration”, in: *Design 2022*, Zagreb, Croatia, http://doi.org/10.1017/pds.2022.202.

Suh, N.P. (2005), “Complexity in engineering”, *CIRP Annals*, Vol. 54 No. 2, pp. 46–63, http://doi.org/10.1016/S0007-8506(07)60019-5.

Vogt, M., Duddeck, F., Wahle, M. and Zimmermann, M. (2018), “Optimizing tolerance to uncertainty in systems design with early-and late-decision variables”, *IMA Journal of Management Mathematics*, Vol. 30, http://doi.org/10.1093/imaman/dpy003.

Zhao, Y.G. and Ono, T. (1999), “A general procedure for first/second-order reliability method (form/sorm)”, *Structural Safety*, Vol. 21 No. 2, pp. 95–112, http://doi.org/10.1016/S0167-4730(99)00008-9.

Zimmermann, M. and von Hoessle, J.E. (2013), “Computing solution spaces for robust design”, *International Journal for Numerical Methods in Engineering*, Vol. 94 No. 3, pp. 290–307, http://doi.org/10.1002/nme.4450.

Zimmermann, M., Königs, S., Niemeyer, C., Fender, J., Zeherbauer, C., Vitale, R. and Wahle, M. (2017), “On the design of large systems subject to uncertainty”, *Journal of Engineering Design*, Vol. 28 No. 4, pp. 233–254, http://doi.org/10.1080/09544828.2017.1303664.