



Stefan Erschen

Optimal Decomposition of High-Dimensional Solution Spaces for Chassis Design

Fachgebiet für Computational Mechanics
Prof. Dr.-Ing. Fabian Duddeck
Technische Universität München



Professur für Computational Mechanics

Optimal Decomposition of High-Dimensional Solution Spaces for Chassis Design

Dipl.-Ing. Univ. Stefan Erschen

Vollständiger Abdruck der von der Ingenieurfacultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Kai-Uwe Bletzinger

Prüfender der Dissertation:

1. Prof. Dr.-Ing. habil. Fabian Duddeck

2. Prof. Dr. Markus Zimmermann

3. Prof. Dr. rer. nat. Matthias Gerdts

Die Dissertation wurde am 22.11.2017 bei der Technischen Universität München eingereicht und durch die Ingenieurfacultät Bau Geo Umwelt am 05.03.2018 angenommen.

Schriftenreihe des Fachgebiets für Computational Mechanics

Band 6

Stefan Erschen

**Optimal Decomposition of High-Dimensional
Solution Spaces for Chassis Design**

Shaker Verlag
Aachen 2018

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: München, Techn. Univ., Diss., 2018

Copyright Shaker Verlag 2018

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-5931-1

ISSN 2193-2700

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: www.shaker.de • e-mail: info@shaker.de

Preface

This thesis is based on the results obtained during my time as a PhD student from 2013 to 2016. During those years I was involved in a joint research project between the Technical University Munich (TUM) and the BMW Group being simultaneously a member of the *TUM Graduate School* and the *BMW ProMotion Program* in Munich. At the TUM, I participated mainly in the activities of the research group of Prof. Dr.-Ing. habil. Fabian Duddeck, Associate Professorship of Computational Mechanics while at BMW, I worked in the development department for driving dynamics, preliminary design.

I would like to express my deep gratitude to my supervising Professor Fabian Duddeck for giving me the opportunity to join his research group, providing me support whenever necessary and enriching my work with a critical review and fruitful discussions.

I would like to thank Prof. Dr. Markus Zimmermann for supervising me at BMW. I particularly appreciate that he gave me the opportunity to do research in an industrial environment, that he shared his expertise in Solution Spaces, in which he has done extensive research in recent years. Furthermore, I would like to point out the numerous valuable discussions we had and still have together.

I also would like to thank Prof. Dr. rer. nat. Matthias Gerdt, Universität der Bundeswehr München, for inspiring discussions and technical support as well as for agreeing to be part of the committee.

Furthermore, I would like to thank Prof. Dr.-Ing. Florian Holzapfel, TUM, for being my mentor during my time as PhD student.

I also would like to thank all of my colleagues at the TUM as well as at BMW for supporting me through helpful discussions, technical support and encouragement. Special thanks to my colleagues, Markus Eichstetter, Daniel Mögele, Martin Münster, Christian Schulz, Marc-Eric Vogt, Jens Wimmer, Martin Wahle and Amir Zare.

I would like to thank my family and friends for proof reading and their valuable comments, and would like to express my deepest gratitude to my father, who enriched this thesis with his admirable sketches.

Last but not least, I would like to emphasize the work done by the numerous interns and Master candidates, special thanks to Thorsten Rauch and Marco Daub. Without their work this thesis would not have been possible.

Stefan Erschen
Munich, 2018

Abstract

Today, one challenge in vehicle development is dealing with increased complexity, i.e. as a consequence of a *large number of interacting components and subsystems*, *many requirements*, often in conflict to each other, and a *large variety of vehicles* that must be considered. In general, complexity leads to time-consuming and hence cost-intensive development processes. In contrast, an increasing number of competitors require efficient product development in terms of time and cost.

Design methodologies such as concurrent engineering and set-based design exist to handle complex design problems more efficiently. One aspect of *concurrent engineering* is that components and subsystems are developed simultaneously rather than subsequently. This potentially reduces the overall development time, however, due to interactions among the components and subsystems, concurrent engineering increases uncertainties due to lack of knowledge. The idea of *set-based design* is to consider a set of permissible designs, which is narrowed throughout the development when more information, e.g. precise customer needs, cost, manufacturability, etc. is available. Although, this requires more effort in early development stages, it minimizes necessary iteration loops due to lack of knowledge and becomes more efficient overall compared to design strategies where one single design is considered only. Set-based design in conjunction with concurrent engineering is a powerful combination of two design strategies, compensating the shortcomings of the individual methodologies and enabling efficient development processes.

Recently, many set-based design approaches, particularly relying on numerical simulation, were proposed to increase the efficiency of development processes. One approach is based on the computation of box-shaped Solution Spaces, representing permissible design alternatives that satisfy all specified requirements. Box-shaped Solution Spaces are compatible to concurrent engineering, since requirements on the system are formulated as independent requirements on components and subsystems, which can be developed in detail independently and simultaneously as a result.

This thesis proposes improved approaches to compute Solution Spaces, such that uncertainties due to lack of knowledge are taken into account at its best, such that the result is compatible to concurrent engineering and such that the approaches are applicable to development problems particularly in chassis design.

Firstly, a gradient-based approach is proposed for optimizing the number of design alternatives via a search of box-shaped Solution Spaces with maximum volume. The underlying optimization problem is analyzed, and the approach is validated via analytic test problems. The results are compared to an existing technique using a stochastic Solution Space algorithm.

Motivated by the fact that box-shaped Solution Spaces are possibly sub-optimal to

represent the whole set of permissible designs, approaches based on a two-dimensional decomposition of high-dimensional Solution Spaces are introduced. This enables an improved handling of uncertainties in the early development phase. Two approaches are presented and the underlying optimization problems are analyzed and discussed. Both approaches are validated again by analytic test problems.

Furthermore, the approaches are compared in terms of their numerical complexity, and the advantages and disadvantages of box-shaped Solution Spaces compared to a two-dimensional decomposition of Solution Spaces are discussed.

Finally, the applicability of the approaches is demonstrated by industrial examples in the field of chassis design. The examples comprise the development of single vehicles as well as the development of a set of vehicles, where the components need to be designed such that requirements on the driving dynamical behavior are satisfied.

Contents

1	Introduction	1
1.1	Context and motivation	2
1.2	Aims of the work	13
1.3	Overview of the methods	14
1.4	Structure of the thesis	14
2	State of the Art	17
2.1	Set-based design	18
2.2	Numerical methods for set-based design	18
2.2.1	Box-shaped Solution Spaces	18
2.2.2	Further interval approaches	19
2.3	Chassis design with box-shaped Solution Spaces	21
2.4	Research questions	23
3	Solution Spaces	25
3.1	Low-dimensional representation of high-dimensional Solution Spaces	26
3.2	Definitions	27
3.2.1	The (complete) Solution Space	27
3.2.2	Parametric vs. oracle functions and surrogates	28
3.2.3	The hull of a Solution Space	29
3.2.4	Size measure of a Solution Space	29
3.2.5	Loss of Solution Space	30
3.2.6	Linearity, monotonicity, convexity	31
3.3	Deriving Solution Space constraints from performance functions	33
3.4	Assessing sets regarding Solution Space constraints	34
3.5	Analytic test problems	34
4	Box-shaped Solution Spaces	39
4.1	Intervals as representation of high-dimensional Solution Spaces	40

4.2	General problem statement	40
4.2.1	Concept of inner and outer box	41
4.2.2	Assessing box-shaped sets regarding Solution Space constraints	41
4.3	Loss of Solution Space	47
4.4	Review: Stochastic algorithm	48
4.4.1	Problem statement	48
4.4.2	Algorithm	49
4.4.3	Properties of the algorithm	49
4.5	Tracking the vertexes of a box: Linearly constrained Solution Spaces	52
4.5.1	Problem statement	52
4.5.2	Implementation	56
4.5.3	Numerical results	57
4.6	Tracking the vertexes of a box: Nonlinearly constrained Solution Spaces	59
4.6.1	Problem statement	59
4.6.2	Implementation	61
4.6.3	Numerical results	62
4.7	Modifications of the optimization problem	65
4.7.1	Implementation	65
4.7.2	Numerical results	66
5	Two-dimensional decomposition of Solution Spaces	69
5.1	2d-spaces as representation of high-dimensional Solution Spaces	70
5.2	General problem statement	70
5.3	Underlying idea	72
5.4	Tracking the vertexes of a polytope	75
5.4.1	Assessing polytope-shaped sets regarding Solution Space constraints	75
5.4.2	Problem statement	77
5.4.3	Implementation	81
5.4.4	Numerical results	82
5.5	Decomposing Solution Space constraints	85
5.5.1	Problem statement	85
5.5.2	Implementation	89
5.5.3	Numerical results	92
6	Comparison of the approaches	95
6.1	Properties of the underlying optimization problems	96
6.2	Numerical effort	98
6.2.1	Box-shaped Solution Spaces: Tracking vertexes of a box vs. the stochastic Solution Space algorithm	99

6.2.2	2d-spaces: Tracking vertexes of a polytope vs. decomposing Solution Space constraints	103
6.2.3	Box-shaped Solution Spaces vs. 2d-spaces	104
6.3	Gain of Solution Space	104
7	Application	107
7.1	Chassis design in an early design stage	108
7.1.1	Design variables and vehicle parameters	108
7.1.2	Performance measures	109
7.1.3	Physical relations between design variables and performance measures	111
7.1.4	Physical simulation model and mathematical surrogates	112
7.1.5	Technical questions	115
7.2	Numerical results of Example One (single vehicles)	118
7.3	Numerical results of Example Two (set of vehicles)	123
8	Critical reflection	131
9	Conclusion	135
A	Optimization	141
A.1	Karush-Kuhn-Tucker conditions for constrained optimization problems . . .	141
A.2	Fundamentals of interior-point algorithms and pattern search	143
A.2.1	Interior-point	143
A.2.2	Pattern search	144
A.3	Number of optimization parameters, optimization constraints and derivatives	147
B	Numerical results	151
B.1	Classification measures	151
B.2	Parametric surrogate models	152
B.3	Monotonicity	153
B.4	Numerical details of the stochastic Solution Space algorithm	158
B.5	Computer specifications	160
	Glossary	169

1 — Introduction

This chapter deals with challenges in the development of complex products, such as passenger vehicles. Existing approaches and methodologies to handle development processes more efficiently are explained. Furthermore, fundamentals such as the decomposition of technical systems, the V-model approach, uncertainties in engineering design, particularly those due to lack of knowledge, apriori information in development, point-based and set-based design methodologies as well as the terms robustness, reliability and flexibility, as used in this thesis, are briefly explained. Subsequently, the aims of the work, an overview of the methods proposed and of the structure of the work is provided.

1.1 Context and motivation

The development of vehicles is complex due to a *large number of interacting components and subsystems, many requirements*, often in conflict to each other, and a *large variety of vehicles* that must be considered. In classical development processes one single design is iteratively modified to satisfy all requirements (called *point-based design*; see subsequent paragraphs). Changes in influencing components or subsystems must be compensated by changes in other components or subsystems and the design process must be repeated, which is time-consuming and hence cost-intensive. However, economical aspects demand development processes to be efficient in terms of time and cost.

Development based on numerical simulation can reduce time and hence cost of development processes and allows vehicle design even if no hardware, e.g. prototypes, subsystem test rigs, etc., is available. Roeski [63] describes a point-based development process for chassis design, where the overall development time is reduced by providing reliable results based on simulation before the first hardware is available. The first phase is solely based on simulation, while the subsequent phase is based on hardware testing, supported by simulation; see Figure 1.1. Thus, hardware testing, which is time-consuming and cost-intensive, can be executed more efficiently. However, particularly in an early design stage, there is only little information about the requirements and conditions, e.g. interacting components, which are expected to change throughout the development process.

Design methodologies such as concurrent engineering and set-based design exist to handle complex design problems more efficiently. One aspect of *concurrent engineering*, see e.g. [58, 59, 76], is that components and subsystems are developed simultaneously rather than subsequently. This allows a reduction of the overall development time, however, in the presence of interactions among the components and subsystems, concurrent engineering increases uncertainties due to lack of knowledge. To deal with lacking information in early design phases, the idea of *set-based design* considers a set of permissible designs, which is narrowed throughout development when more information, e.g. precise technical details about interacting components, cost, manufacturability, etc. is available. Although, this requires more effort in early development stages, it minimizes necessary iteration loops due to lack of knowledge and becomes more efficient overall compared to design strategies where one single design is considered only. Set-based concurrent engineering, see e.g. [2, 70], is a powerful combination of the aforementioned design strategies and enables taking uncertainties into account and handling complex design problems more efficiently.

Recently, many set-based design approaches, which particularly rely on numerical simulation, were proposed to increase the efficiency of development processes. Zimmermann et al. [78] propose a stochastic algorithm based on numerical simulation to compute box-

shaped Solution Spaces. Rather than one design, Solution Spaces provide a set of designs that is considered throughout the development process and narrows when more and more information is available. Subsequently, the most promising design out of the set of remaining designs is realized. Inside the Solution Spaces, all requirements on system response are satisfied. Designs within the Solution Space are called good designs. Time-consuming iterations can be reduced by providing space for variation. Box-shaped Solution Spaces are compatible to concurrent engineering, since requirements on the system are formulated as independent requirements on components and subsystems (expressed as permissible intervals), which can subsequently be developed in detail independently and simultaneously. Thus, box-shaped Solution Spaces can be considered as a set-based concurrent engineering approach, enabling efficient development processes. However, box-shaped Solution Spaces are - depending on the problem - ill-suited to approximate the entire set of good designs, and the computation can be computationally expensive.

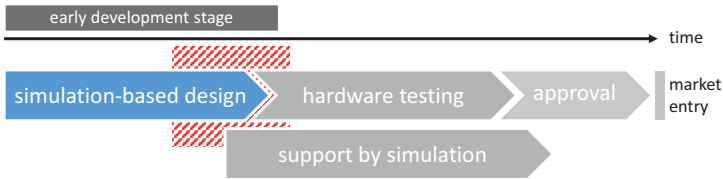


Figure 1.1: Development process with multiple stages, the first stage is based on numerical simulation and the subsequent stage is based on hardware testing, taken from [63]. The red shaded area highlights the phase, which is mainly discussed in this thesis.

The methods proposed in this thesis can be used in all stages of development where numerical simulation is applied to design vehicles. The examples considered here focus on a stage, where conceptual decisions have been already made. This stage is characterized by the fact that decisions about the type of the vehicle (sedan, coupé, etc.), the type of engine (combustion engine, electric engine, etc.), the drive concept (rear-wheel drive, front-wheel drive, all-wheel drive), etc. have been already made. Typical components in chassis design that are developed in such a stage are the anti-roll bars, coil springs, dampers, bump stops, rebound stops (front and rear axle, respectively), tires, etc; see component level in Figure 1.2. The design variables are the properties of the components, such as the stiffness and length. The system is often evaluated by considering performance measures, i.e. quantitative measures such as the roll angle of a vehicle while executing a predefined maneuver. Although much of the system is known, there is still a high number

of possibilities for realizing the vehicle. To illustrate this, consider an example where two components of a system satisfying a particular requirement must be designed, taking into account that each component can be replaced by five different parts. In this case, one must evaluate $5^2 = 25$ system configurations. The effort for the evaluation of all possible configurations increases significantly with the *number of components* that must be designed, the *number of different alternatives/parts for each component*, the *number of requirements on the vehicle* and the *number of vehicles* that must be developed. A chassis design problem comprising five components and 20 alternatives for each component, six requirements and 20 different vehicles, yields a number of $20^5 \times 20 = 64,000,000$ possible configurations, each needs to be evaluated w.r.t. six requirements. The evaluation of all possible configurations is impracticable with simulation and impossible with hardware testing. With an industrial example in Chapter 7 it is shown how the method of Solution Spaces can be used to handle problems of this size while taking uncertainties due to lack of knowledge into account.

Decomposing a system into subsystems and components Each system (vehicle) can be decomposed into subsystems (axle, engine, body, etc.) and components (anti-roll bar, bump stop, rebound stop, etc.); see Figure 1.2. Some components can be further decomposed, e.g. the damper is an assembly of i.a. a piston, a piston rod and a damper tube. The system and each subsystem and component are characterized by multiple properties, such as cost, mass, etc. Some of these properties show interactions among each other that can be visualized by a graph. The graph is arranged as follows: The top level of the graph shows properties of the system, the lower levels show properties of its subsystems and components. Arrows indicate direct relations between different properties. Figure 1.3 depicts an example from chassis design. The focus in this thesis is on particular properties of the components anti-roll bar of the front and rear axle (stiffness), bump stops of the front and rear axle (stiffness and length), rebound stops of the front axle (stiffness and length), as well as on particular properties of the overall vehicle (particular performance measures for driving dynamics, influenced by the overall mass and inertia around the vertical axis as well as the position of the center of gravity in vertical and longitudinal direction of the vehicle).

The relations among the properties can be quantified by physical experiments or mathematically. In the latter case, the system's response (= a property of the system), for instance, can be expressed by a function, in the following called performance function. It maps the value of a design variable (or the values of several design variables), in the following the property of a component (or several properties of several components), denoted by x , onto a value of a system's response, in the following called performance measure, denoted by z , i.e. $f : x \mapsto z$.

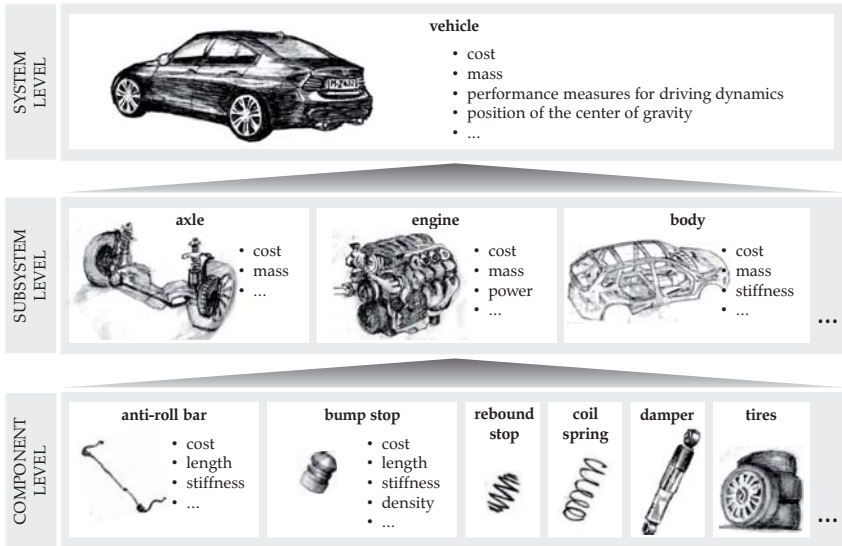


Figure 1.2: Each system can be decomposed into subsystems and components. The system and each subsystem and component is characterized by properties, such as cost, mass, etc.

Development with the V-model approach The V-model approach, see [38], is often used to handle complex product development projects. While its origin is the software industry, today, in the framework of systems engineering, it is an established design process philosophy in many other industries, such as the automotive industry; see [79]. The main idea is to break down requirements from a level, where the overall product or system is considered, to one or more levels, where details of the product are taken into account. Therefore, this part of the approach is called *top-down approach*, i.e. from the system level (top) to a detail level, e.g. subsystem or component level (bottom). If all details are specified, the system is assembled and the overall system performance is validated, called *bottom-up*, i.e. from a detail level (bottom) to the system level (top).

Consider the following *simple example from chassis design* with one requirement on the system's response and one property of a component that must be designed: On the system level a requirement in terms of a maximum roll angle of the vehicle while cornering is given. In accordance with the V-model approach this requirement is broken down to the subsystem level in terms of a requirement on the minimum vertical stiffness of the front and rear axle, respectively. This can be further broken down to a more detailed level, let's say a minimum stiffness of the anti-roll bar. If the anti-roll bar is designed

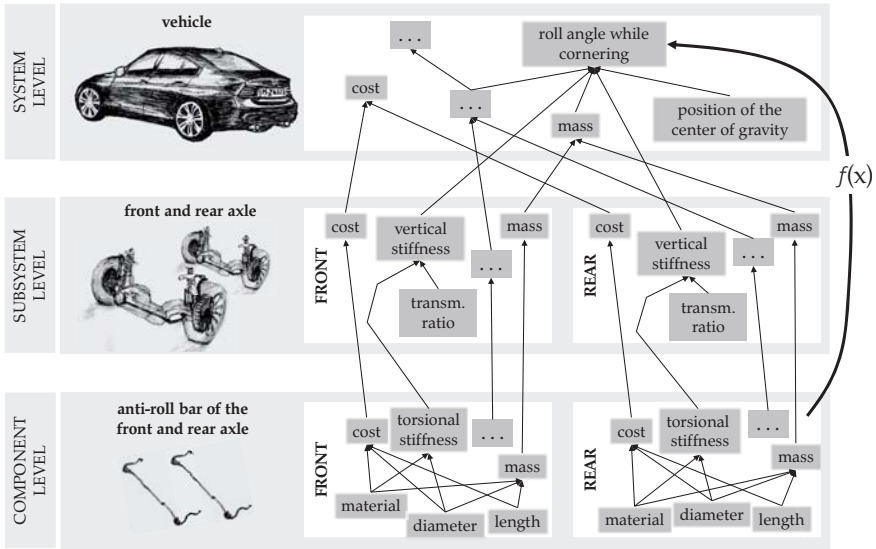


Figure 1.3: The graph shows the dependencies of different properties of components, subsystems and the system. The dependencies can be quantified by a performance function $f(x)$, mapping the values of particular design variables x (here properties of components) onto a value of a performance measure of the system.

and realized, the system is assembled and the overall system performance is validated. While this example is trivial, consider a complex product with hundreds of interacting properties of components and subsystems and thousands of details that must be designed. The V-model approach enables the development of complex products to be accomplished by several groups in a structured and efficient manner.

Uncertainties in engineering design In literature there are many definitions and attempts to categorize uncertainties which arise during development or when the product is in service. Often uncertainties are categorized as **aleatoric** and **epistemic**; see e.g. [21]. The word *aleatoric* is derived from the Greek word *alea*, which means rolling of dice and implies that this type of uncertainty occurs due to randomness. Examples are scatter in material properties due to imprecise manufacturing processes, variation in road surface conditions, etc. *Episteme* means knowledge and epistemic uncertainties are those caused by lack of knowledge, e.g. the cost of particular components, which is often not known in early development stages. A different categorization uses the terms **controllable** and **uncontrollable factors**, see e.g. [22, 41, 51]. While controllable factors are those properties

of the system that can be designed (design variables), uncontrollable factors are environmental conditions, such as applied loading, aging, temperature, etc. In the framework of distributed development, see concurrent engineering, all properties, which are designed in other departments and hence cannot be influenced by the responsible design team, are uncontrollable factors (e.g. the vehicle parameters in the application example in Chapter 7). Padulo [55] distinguishes between uncertainty **about the problem** (undefined scope of the problem, uncertainties introduced by simplification, etc.) and **within the problem** (incomplete data, uncertain model structure, model parameters, etc.). Further explanations and approaches for modeling uncertainties are given in [12].

To incorporate uncertainty evaluation into the development and hence avoid undesired system behavior, uncertainties must be quantified. Therefore, uncertainties are normally specified by particular probability distributions, e.g. Gaussian distribution, and statistical values such as mean and standard deviation. Particularly epistemic uncertainties, where mostly nothing is known about the distribution, are often modeled by uniform distributions, i.e. all values within a particular range are assumed to be of the same likelihood. More advanced methods use possibilistic and fuzzy approaches, e.g. [48].

Many approaches based on numerical simulation have been proposed in literature to take uncertainties into account, e.g. robust optimization methodologies [41]. Often the approaches proposed can deal with a particular type of uncertainty only, e.g. model inaccuracies, tolerances in manufacturing, etc. However, uncertainties due to lack of knowledge, for instance, require other approaches. This is discussed in the following.

Uncertainties due to lack of knowledge Particularly in early design stages only little is known about the system that needs to be designed. In this case, **lack of knowledge** means uncontrollable variations in the properties of the system, the subsystems or components as well as in requirements on the system. In the following, two typical situations, which arise during development of complex products in early stages, are mentioned.

Situation A: Designing components such that system requirements are satisfied, however, influencing *components or subsystems are unknown or show uncontrollable variation*; see Figure 1.4 left. Possible reasons are:

1. *Distributed development processes:* The development of complex products, such as vehicles, is normally split up into different groups; see Figure 1.5 left. Although components and subsystems are highly coupled and as a whole influence the performance of the system, they are often developed simultaneously; see concurrent engineering.
2. *Changes throughout the development process:* In early development stages, components and subsystems are not designed in detail, and properties change while more and more details are specified.

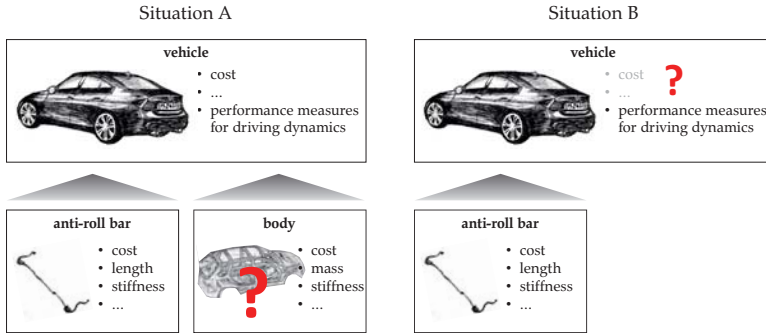


Figure 1.4: Typical uncertainties due to lack of knowledge during the development of complex products in early stages. *Situation A* (left): A component (e.g. anti-roll bar) must be designed but properties of influencing components and subsystems (e.g. body) are unknown or show uncontrollable variation. *Situation B* (right): A component (e.g. anti-roll bar) must be designed but not all requirements are known or not all requirements are considered.

For instance, consider the situation of developing the anti-roll bar of a vehicle, while the mass of the vehicle is uncertain. Both, the anti-roll bar and the mass, have influence on crucial performance measures in the field of safety and comfort and thus, the anti-roll bar must be designed in accordance with the mass. Possible scenarios that yield uncertainties in the mass of the vehicle are: Other subsystems, e.g. the body of the vehicle is designed by other groups and is not specified yet (*Situation A(1.)*) or is not designed in detail and hence is expected to change (*Situation A(2.)*).

Situation B: Designing properties of components such that system requirements are satisfied, however, *not all requirements are known or not all requirements can be considered*; see Figure 1.4 right. Possible reasons are:

1. *Distributed development processes:* One component or subsystem often influences many properties of the system, but often not all requirements on the system are considered when designing a component. The reason is that targets are often assigned to different groups (e.g. driving dynamics vs. cost; see Figure 1.5 right) and each group is responsible for the associated targets only. If a component or subsystem is designed such that the system satisfies a particular requirement, it may fail regarding other requirements, if they were not taken into account.
2. *Limited possibility to assess particular properties:* Particularly in an early design

stage, hardware is not available and instead numerical simulation models are used. Often simplified models are used in order to reduce the computational effort or when there is insufficient information to build up detailed models of a system. This often implies that some phenomena and hence requirements on properties cannot be assessed in an early stage.

Situation A and *Situation B* mean to develop under uncertainty, which must be quantified, if possible, and taken into account.

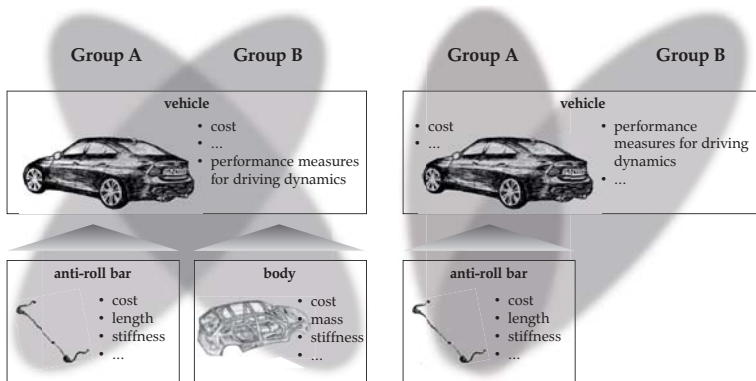


Figure 1.5: Two types of distributed development processes: Either each group is responsible for a particular component or subsystem (left) (leads to *Situation A*: influencing components or subsystems show uncontrollable variation) or each group is responsible for a particular target (right) e.g. driving dynamics vs. cost (leads to *Situation B*: not all requirements are considered at once).

Apriori information in development Even in early development stages, apriori information is often available when designing products. Examples are the knowledge that some design variables require more space for variation than others due to imprecise manufacturing processes or the knowledge that some components are expected to scatter in a particular range. This knowledge is often based on the experience of experts. Another example is the substitution of components due to product families. A product family is often defined as a set of products sharing the same basic platform with the possibility to create a portfolio of diverse products by substituting components and subsystems; see [66]. Therefore, a system must be designed such that the requirements on the system are satisfied even if some components or subsystems are substituted by others; see Figure 1.6. An example from the automotive industry is a vehicle that is planned to be equipped with

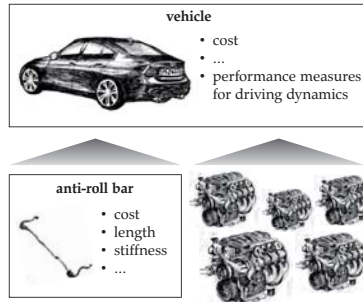


Figure 1.6: Properties of components (e.g. anti-roll bar) must be designed, such that requirements on the complete vehicle are satisfied even if some components or subsystems (e.g. combustion engines) are modified or substituted.

different engines (gasoline 3-cylinder, gasoline 4-cylinder, diesel 4-cylinder, etc.), however, the remaining components and subsystems are the same.

Point-based vs. set-based design Singer et al. [68] summarize and compare the main ideas of point-based and set-based approaches. **Point-based design** is characterized as (in accordance with [46]): *One single design* is developed throughout the development process and *modified in each development step* such that it *satisfies more and more specifications/requirements*. In the case that it turns out that the design is not able to satisfy all specifications, the process must be repeated; see Figure 1.7. The result is often a sub-optimal design, as decisions made in an early design stage - based on the little information available - may have prevented a better design.

Set-based design is characterized as: *Many different design alternatives (sets of permissible designs) are considered throughout the development process*, and designs are eliminated only if enough information is available, i.e. with increasing information the *set of designs narrows throughout the development process*, and a *single design is chosen at the end*.

Figure 1.8 illustrates the idea of set-based design for a design problem with two and more design variables. The set of permissible designs shrinks during the development process when more and more requirements are taken into account. Finally, a single design is chosen e.g. one that can be manufactured with minimum cost. While the visualization of the set of permissible designs is possible in two dimensions the visualization for more than two dimensions needs special techniques, e.g. box-shaped Solution Spaces (intervals

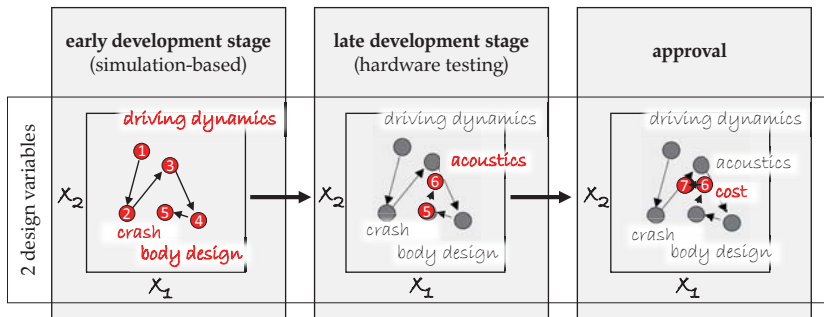


Figure 1.7: The example illustrates a design problem with two design variables, denoted by x_1, x_2 . Point-based design is characterized by many design changes and hence iterations during the development process, necessary when more and more requirements are taken into account.

for each design variable) or 2d-spaces (target regions for pairs of design variables). In set-based design as many bad designs as possible are excluded in early development stages in order to reduce the effort for unnecessary evaluations in subsequent stages; see [61]. Note that Solution Spaces are in accordance with set-based design: By considering as many requirements as possible in early stages the size of the complete Solution Space decreases. In contrast to the existing set-based design methods, however, the presented methods here aim at keeping many permissible designs by maximizing a Solution Space (e.g. a box-shaped Solution Space). The motivation is to capture as many good designs of the complete Solution Space as possible and consequently, to provide maximum space for variation due to lack of knowledge.

For examples of point-based and set-based design methodologies see Figure 1.9. While classical optimization seeks one single design with the best (here minimum) performance (e.g. energy consumption, noise, cost), robust design optimization seeks a design with the best performance in conjunction with low variation in the performance for a given probability distribution of the design variable values. Reliability design optimization also considers a given probability distribution of the design variable values but seeks a design with the best performance that satisfies the requirement (constraints) with a desired probability (= probability of failure, e.g. 10^{-9}). The computation of Solution Spaces however, provides a domain in the input space (= space of design variables) rather than a single design, in which the system satisfies the requirements, given in terms of thresholds w.r.t. performance measures.

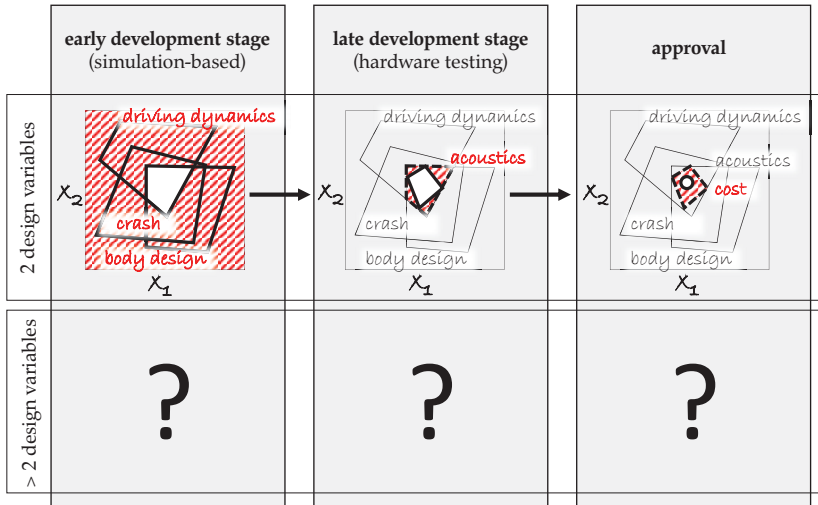


Figure 1.8: The idea of set-based design for a design problem with two design variables, denoted by x_1, x_2 . While more and more requirements are considered, e.g. requirements on crash behavior, driving dynamics, body design, etc., the set of design alternatives (white area) narrows throughout the development process and a single design (black circle) is realized at the end. Whereas the set of permissible designs can be illustrated easily for a two-dimensional problem, high-dimensional problems require special techniques.

Robustness, reliability and flexibility Since there are many different definitions for **robustness** (e.g. robust design optimization [12]), **reliability** (e.g. reliability-based optimization [25]) and **flexibility** (see [64]) in the framework of engineering system design, the meaning of these terms in the context of Solution Spaces is explained briefly. Robustness, reliability and flexibility are synonyms in this context, i.e. a robust/reliable system yields flexibility, and a system which yields high flexibility is also robust and reliable. Here, a system is called robust or reliable, if properties of subsystems or components may change within a particular range, while the system still satisfies all requirements. The size of the set of permissible designs can be seen as a measure for robustness and reliability respectively, since the larger the set of permissible designs the lesser the probability of the system to fail if some properties vary randomly, e.g. due to lack of knowledge. Or in other words, the larger the size of the set of permissible designs, the more possibilities for a decision maker from which he or she can select and hence the greater the flexibility. Here, the term flexibility refers to the development process not to the system, this is in accordance with Chen [18]: *“Our aim is to provide flexibility in the design process [...]”*.

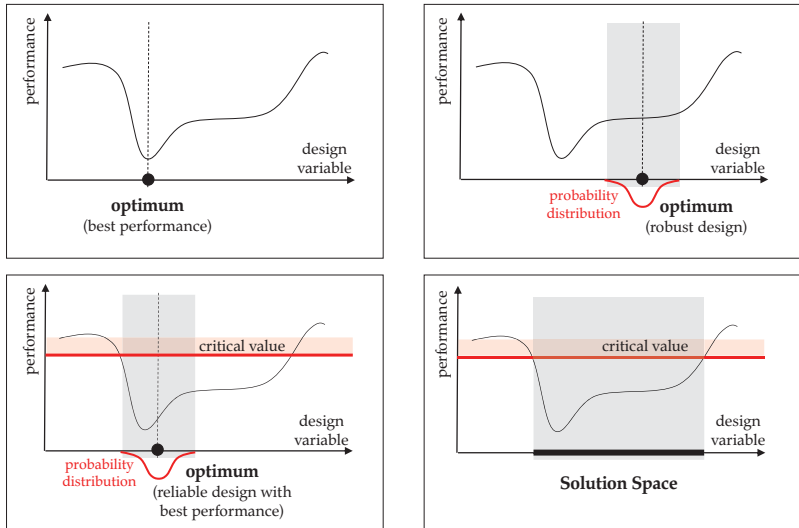


Figure 1.9: Whereas classical optimization (top left), robust design optimization (top right) and reliability design optimization (bottom left) yield one single design, Solution Space approaches (bottom right) provide a set of designs.

By flexibility we mean that instead of looking for a single point solution in one disciplines model, we look for a range of solutions that involve information passing between multiple players (disciplines)”. For a discussion of the term flexibility in the context of decision theory and engineering design see [64].

1.2 Aims of the work

Large Solution Spaces provide space for variation due to lack of knowledge, yield flexibility for decision makers in subsequent design stages and consequently avoid time-consuming and cost-intensive iterations in the development process.

The *first aim* of this thesis is to develop efficient methods for the computation of box-shaped Solution Spaces. This is achieved by extending the *direct approach* of Fender [28, 29], where the problem of seeking box-shaped Solution Spaces is solved analytically or by applying standard optimization algorithms to general problems with application in chassis design.

When considering industrial problems, box-shaped Solution Spaces are often ill-suited to capture the entire set of good designs. Consequently, the *second aim* of this thesis is to

seek alternatives to the established box-shaped Solution Spaces that are able to provide more good designs and hence, provide more robustness and flexibility. An alternative approach is introduced, where high-dimensional Solution Spaces are represented by a set of independent two-dimensional Solution Spaces, called 2d-spaces rather than intervals.

Often, apriori knowledge of developers, e.g. a desired variation in influencing properties due to product family design, is available when designing products. Consequently, the *third aim* is to adapt the approaches such that this apriori knowledge can be taken into account. Therefore, modifications of the underlying problem statements are proposed.

In summary the aims of this work are:

- *Aim 1*: Providing alternative methods for efficient computation of box-shaped Solution Spaces.
- *Aim 2*: Developing alternatives which are able to provide more good designs compared to box-shaped Solution Spaces in order to increase the flexibility and robustness w.r.t. lack of knowledge in development processes.
- *Aim 3*: Adapting the approaches such that apriori knowledge can be considered in order to make the method of Solution Spaces applicable to a wider range of typical development questions in the field of chassis design.

1.3 Overview of the methods

In this thesis, problem statements for solving the problem of seeking Solution Spaces with maximum size are presented. On the one hand, a stochastic algorithm to compute box-shaped Solution Spaces, introduced in [78], is reviewed. On the other hand, problem statements are presented that allow to solve the problem with standard optimization algorithms. Additionally to the box-shaped Solution Spaces, the idea of a two-dimensional decomposition of a high-dimensional Solution Space is introduced. To compute a Solution Space with maximum size, represented by two-dimensional Solution Spaces (2d-spaces), two methodologies are provided. To solve the problems with maximum efficiency, a gradient-based optimization algorithm is used whenever gradients of the problem can be derived by analytic considerations. An overview is given in Figure 1.10.

1.4 Structure of the thesis

The following chapters begin by providing an excerpt of past as well as ongoing research in related fields and continue by stating fundamentals about Solution Spaces. They offer proposals on optimization problems for computing optimal Solution Spaces along with a

	stochastic Solution Space algorithm	standard optimization algorithms
box-shaped Solution Spaces	U1 stochastic Solution Space algorithm for box-shaped Solution Spaces	V1 tracking vertexes of a box
2d-spaces		W1 tracking vertexes of a polytope W2 decomposing Solution Space constraints

Figure 1.10: Overview of the method reviewed ($U1$) and the methods proposed in this thesis ($V1$, $W1$ and $W2$). The rows distinguish between the different types of Solution Spaces, i.e. box-shaped and 2d-spaces and the columns between the different methods for the computation of Solution Spaces, either by a stochastic Solution Space algorithm (reviewed) or by applying standard optimization algorithms.

comparison of the results for analytic examples. They conclude with an application of the proposed methods to a real world chassis design problem.

In particular the chapters comprise the following contents:

- In *Chapter 1* challenges arising in development processes for designing complex technical products are explained and state-of-the-art methodologies to cope with these challenges are briefly described. Furthermore, fundamentals are explained, the aim of the work is stated and an overview of the approaches proposed in this thesis is provided.
- In *Chapter 2* publications about set-based design are summarized and discussed, followed by numerical approaches which allow to compute sets of designs. Finally, the findings of applying a Solution Space approach to chassis design problems are summarized and the research questions are derived.
- In *Chapter 3* fundamentals of Solution Spaces and definitions are stated. Addition-

ally, test problems are proposed, which are used for the analysis of the developed approaches.

- In *Chapter 4* an existing approach for the computation of box-shaped Solution Spaces, based on a stochastic algorithm, is reviewed. The results of an analysis of the numerical effort of the algorithm are briefly summarized. Furthermore, an approach where only the vertex or particular vertexes of the box are considered in order to find a box with maximum volume that satisfies all requirements is proposed. The mathematical conditions for the validity of the approach are presented and numerical results based on the test problems are provided and discussed. At the end of the chapter, modifications of the introduced problem statement are presented, which allows to provide an answer to a wider range of typical questions arising in chassis design.
- In *Chapter 5* a new approach to represent high-dimensional Solution Spaces is presented. Therefore, Solution Spaces are decomposed into a set of two-dimensional Solution Spaces. Two ideas as well as the associated underlying optimization problems are stated and numerical results are provided and discussed.
- In *Chapter 6* the underlying optimization problems of the proposed approaches are analyzed and compared. Additionally, pros and cons of box-shaped Solution Spaces compared to a two-dimensional decomposition are discussed and the advantage of 2d-spaces in terms of gain of Solution Space is shown by an analytic example.
- In *Chapter 7* the approaches are applied to a chassis design example to demonstrate the applicability by answering typical questions arising in chassis design in early stages.
- *Chapter 8* refers back to the aims of the thesis and critically reflects the results.
- *Chapter 9* summarizes the results, points out the main conclusions of the work presented and provides an outlook for future research.

2 — State of the Art

In the previous chapter set-based design was introduced as a reasonable approach in development of complex products in order to enable efficient and robust development processes. In this chapter, existing set-based design methodologies are summarized and discussed. Furthermore, recent approaches for the computation of sets of designs expressed as intervals are evaluated, followed by publications about the application of box-shaped Solution Spaces for chassis design. Finally, the necessity for new approaches, presented in this thesis, is pointed out.

2.1 Set-based design

The term *set-based design* was mainly formed by Ward [73] in 1989 and appeared in many further publications in the following years, e.g. in [11,31,32,32,46,61,68–70,72,74,75]. Set-based design in conjunction with concurrent engineering, i.e. simultaneous development of subsystems or components by different teams, was discussed in [70,74] and [51].

Nahm et al. [51] describe the principles of set-based concurrent engineering by establishing three phases: In phase one, each team defines permissible regions in the design space, i.e. possible designs satisfying the individual requirements under consideration. In phase two, each team incorporates the information of the other teams and a set of designs that satisfies all requirements is sought. While more and more details are considered throughout the development process further designs are excluded and a final design is selected at the end (phase three); see Figure 1.8.

Rekuc et al. [61] point out that, for an efficient application of set-based design methodologies, it is crucial to eliminate as many designs as possible in early development stages. The reason is that the evaluation of design alternatives gets more and more expensive in terms of time and cost and reaches its maximum at the end of the development. The authors propose an approach based on intervals to eliminate design alternatives even in case of imprecise knowledge. An extension of this approach is presented in [56].

2.2 Numerical methods for set-based design

In the literature, several approaches, that seek intervals for each of the design variables rather than a single design can be found. All these approaches have in common that they map permissible ranges of the system's outputs, i.e. requirements on performance measures, onto permissible ranges of the system's inputs, i.e. intervals for each of the design variables. The Cartesian product of all intervals is an axis-parallel hypercube, in the following called **box**, in the **input space**, i.e. space of design variables.

2.2.1 Box-shaped Solution Spaces

Zimmermann et al. [78] introduce a stochastic algorithm for the computation of box-shaped Solution Spaces based on Monte Carlo sampling and Bayesian statistics. The algorithm starts with a Monte Carlo sampling within a specified start box in the input space, i.e. the space of design variables, followed by function evaluation of each sample point, verification if the performance of each sample point is beyond the specified threshold, categorization of the points as good or bad designs and application of a trim algorithm for elimination of bad designs in the box. Subsequently, the box is expanded. The process is repeated such

that the box moves towards an area with more good designs. The algorithm can handle any nonlinear, high-dimensional and noisy problem, however, it requires many function evaluations; see also [36,37,45]. In [30] the algorithm is extended by additional constraints, which allows to turn a bad design into a good design with a minimum number of design variables to be modified. The stochastic algorithm is reviewed in Section 4.4.

Fender [28, 29] presents a new approach for the calculation of box-shaped Solution Spaces particularly for front-crash structural design. The complete Solution Space is described by a set of linear inequalities, which are derived from a crash model for an USNCAP full frontal crash load case based on discrete masses and elements. Within the permissible domain, the minimum interval width is maximized, which yields a linear objective function. This problem can be solved directly by means of Lagrange multipliers. In presence of objective functions of higher than second order or more complex systems, for which the analytic computation becomes more and more difficult, the problem can be solved by numerical optimization. The result is a box that strictly fulfills all requirements. Fender mentions the idea of increasing the width of the intervals by moving one vertex of the optimal box into the domain of bad designs, while the fraction of good designs remains almost 100%. The latter is discussed in more detail in [36] and is related to the *Vertex Problem* briefly described later.

2.2.2 Further interval approaches

Fung et al. [34] introduce an approach to compute a set of permissible intervals for design variables using a support vector machine or any other hyperplane-based linear classifier. This is achieved by solving a set of constrained optimization problems and motivated by the fact that intervals are more intuitive than black-box classifiers. The performance of the so-called rule extraction algorithm is demonstrated by a medical example. In order to describe the feasible domain as completely as possible, many non-overlapping hypercubes are generated. Although the problem solved is similar to that discussed in [28, 29] it is different in the sense that it is only applicable in the presence of one linear inequality, i.e. the hyperplane.

Beer et al. [10] propose a methodology for computing intervals for each design variable relying on cluster analysis incorporating fuzziness. Therefore, sample points are generated within the input space and cluster analysis is applied to those sample points that satisfy all requirements on the system. Based on these clusters, i.e. discrete sets of sample points, box-shaped domains (hypercubes) are computed by an iterative approach, which successively reduces the size of the domain until the domain contains permissible points only. Each box-shaped domain is assessed regarding particular criteria, e.g. robustness and volume, and a decision maker can select in accordance with his or her preferences. The approach relies on a single set of sample points within the input space and hence

provides, on the one hand, reliable results even for non-connected permissible ranges. On the other hand, in the presence of many dimensions, this yields inaccurate results or a high computational effort due to the necessity of a large number of sample points. The approach is demonstrated on a five-dimensional structural design problem in the field of vehicle crashworthiness.

In order to deal with lack of information in early design stages, Götz et al. [35] propose an algorithm similar to that published in [10]. Based on computed sets of permissible designs obtained by applying cluster analysis, hypercubes with maximum volume or largest possible minimum interval width are sought. Therefore, different approaches are explained, which differ in the number of degrees of freedom and thus, complexity. One of the approaches, the interval approach, is based on the computation of sub-hypercubes. The approach is demonstrated by an eight-dimensional example in the field of vehicle crashworthiness.

To provide the maximum possible variability of input parameters such that the system output is within a specified range, Rocco et al. [62] propose an algorithm where the optimization problem of seeking a hypercube with maximum volume is solved by cellular evolutionary strategies and the evaluation of the hypercube is accomplished by applying interval arithmetic. It is shown that the approach is able to handle nonlinear problems with concave permissible domains. However, applying interval arithmetic has two drawbacks. Firstly, the output functions must be given in an analytic form and secondly, the output may be overestimated, see [39], and as a consequence the obtained box does not have maximum size, contains infeasible designs or no box is found.

Nahm et al. [51] propose a methodology for an implementation of set-based concurrent-engineering, comprising a space representation method, a space mapping method and a space narrowing method.

- Space representation method: The designer's preferences on the inputs, i.e. design variables, as well as on the outputs, i.e. performance measures of the system, are specified by preference functions, similar to membership functions in the framework of fuzzy sets. This allows to specify not only one single possible design space and required performance space, i.e. design requirements, but a whole set of design spaces and desired performance spaces and thus flexibility within the development process.
- Space mapping method: To map intervals of design variables into the space of performance measures, the Interval Propagation Theorem [31, 32], which is valid for continuous functions, monotone with respect to each variable, is applied.
- Space narrowing method: By applying design of experiment techniques in an iterative procedure the initial design space is narrowed such that unacceptable sub-

spaces are excluded. To select the subspace that fits best to the user's preferences and desired robustness the preference and robustness indices are introduced. The approach is demonstrated by a crash problem with two design variables and five performance measures. The relations between inputs and outputs are approximated by a quadratic model, fitted to data obtained from FEM simulations.

2.3 Chassis design with box-shaped Solution Spaces

Note that in the framework of Solution Spaces the designs are solely characterized whether they satisfy the requirements on the system, in the following called good designs, or not, in the following called bad designs. Zimmermann et al. [78] point out the following advantages of seeking intervals rather than a single design:

- Requirements on the system level, provided as lower or upper thresholds w.r.t. some properties of the system, can be expressed as permissible intervals on the component level, which enables to develop in accordance with the V-model approach. The intervals on the component level are independent of each other and hence enable distributed development processes where different groups are able to develop components in detail independently and simultaneously. Consider Figure 1.3: For instance, a requirement on the roll angle while cornering is broken down on the torsional stiffness of the anti-roll bar of the front axle and on the torsional stiffness of the anti-roll bar of the rear axle. The geometrical properties such as the diameter then can be selected for each component independently.
- Intervals provide space for uncertainties, in particular for uncertainties which arise from lack of knowledge and hence cannot be described by a specified probability distribution. The volume of the box as size measure of the set of good designs is a measure for robustness and flexibility; see paragraph *Robustness, reliability and flexibility* in Section 1.1.
- Intervals enable the visualization of a high-dimensional set of good designs in an easily understandable manner.

Eichstetter et al. [24] apply the stochastic algorithm, introduced in [78], to compute Solution Spaces for a design problem in chassis development. A box-shaped Solution Space is sought for the parameters of a force-velocity characteristic of a damper, which encloses those designs that satisfy the requirements on comfort and lateral dynamics. The authors point out that the approach provides flexibility to consider further requirements on the system in a later development phase and hence avoids time consuming loops. In [23], Eichstetter et al. demonstrate the ability of this approach to design systems with optimal

commonality, i.e. particular components are shared by diverse systems (= product family design). Intervals for six components of 13 vehicles are computed separately and overlapping regions are determined by the proposed algorithm. Eichstetter [22] describes how the method of Solution Spaces can be used in chassis design to improve the development process in terms of handling complexity, involving aspects of robustness, reducing effort and enabling a better system design. The computation of Solution Spaces requires objective system targets, i.e. performance measures in conjunction with thresholds. However, in industry, system targets are often formulated as subjective targets. Therefore, the author describes a methodology to derive objective system targets on the basis of subjective requirements. Furthermore, he explains the contribution of Solution Spaces to a robust design process and introduces an algorithm for designing product families with Solution Spaces. The stochastic algorithm is used to find box-shaped Solution Spaces for a real world problem with ten design variables under consideration of six requirements on five performance measures from driving dynamics. In addition to [78], Eichstetter mentions the following advantages of Solution Spaces, particularly in chassis design:

- Further requirements can be considered in a later phase without causing a redesign of the system by consideration of additional box-shaped Solution Spaces. For instance, assume that all designs within a box satisfy particular requirements and all designs within another box satisfy another set of requirements. The overlap of both boxes are those designs that satisfy all requirements under consideration. This relates to lack of knowledge uncertainties; see *Situation B* in Figure 1.4.
- A reliable design can be identified easily as the one in the center of the Solution Space, since it is the design with the maximum distance to the edges of the box, i.e. to those designs which might fail w.r.t. one of the requirements.
- The design in the center of the box allows maximum variability in case of uncertain design variables.
- Robustness is enabled in the sense that the variation of an uncontrollable parameter, expressed as lower and upper bound, can be considered by an additional interval, which is set fix during optimization.
- A product family can be designed with improved commonality, by superposition of several box-shaped Solution Spaces. Therefore, Solution Spaces are computed for different vehicles individually and overlapping regions indicate components that can be shared.

Further applications of the method are provided in [50] and [77]. In [50], intervals for seven design variables of a vehicle steering design problem are computed and in [77], tires

are designed in accordance with the axle of a vehicle. The results of the application of box-shaped Solution Spaces in crash and chassis design are summarized in [80].

2.4 Research questions

The stochastic algorithm proposed in [78] to compute box-shaped Solution Spaces can handle arbitrary design problems; however, in presence of high-dimensional problems it requires many function evaluations. Graff et al. [37] state that the speed of convergence of the algorithm proposed in [78] decreases with an increasing number of dimensions. Fender [28, 29] shows that in case of a particular crash problem a different approach applying standard optimization algorithms is able to compute box-shaped Solution Spaces more efficiently. Other approaches to compute intervals, which are able to handle one single linear constraint, see [34], or rely on the information of a single set of sample points within the input space, see [10, 35] can be found in literature. For a sufficient exploration of the input space, sample based approaches in general require many function evaluations, due to curse of dimensionality; see [42].

Chassis design problems often comprise many design variables. Additionally, in chassis design often a large number of box-shaped Solution Spaces must be computed, e.g. to enable product family design; see [22, 23]. Furthermore, the performance measures are generally determined by numerical black-box simulation, and hence analytic functions are not available. Chassis design problems also comprise many requirements on performance measures, and hence, multiple constraints must be considered. All this requires searching for efficient methods, particularly applicable to chassis design problems.

Box-shaped Solution Spaces enable independent design work, since requirements on the system are formulated as individual requirements on each of the design variables; see V-model approach. Thus, components or subsystems can be developed in detail independently. The drawback, however, is that box-shaped Solution Spaces are, due to geometrical mismatch, generally not able to capture all good designs. In chassis design, often at least two design variables are assigned to one component, and hence a full decoupling is not necessary. Taking this into account, alternative techniques can be developed that spare to decouple requirements for each of the design variables individually, while providing larger sets of good designs.

In chassis design, often a priori information is available. For instance, particular design variables need more space for variation, some design variables scatter in a particular range, etc. For application in industry it is essential that numerical methods can take this into account.

Based on these findings, the following questions arise and will be answered in this thesis:

- How can box-shaped Solution Spaces with application in chassis design be computed more efficiently compared to existing approaches in terms of computational cost?
- How can the set of permissible designs be represented such that the loss of Solution Space is minimum compared to box-shaped Solution Spaces, whereas components can still be developed independently?
- How can the approach of Solution Spaces be extended to incorporate apriori knowledge and hence make it applicable to further questions arising in development?

In this thesis, the approach introduced in [28,29] is extended to enable the efficient computation of box-shaped Solution Spaces in chassis design by applying standard optimization algorithms. Furthermore, the approach of 2d-spaces is proposed, which is different compared to box-shaped Solution Spaces such that the set of good designs is represented by two-dimensional target regions rather than intervals. Every 2d-space shows permissible regions for a predefined pair of design variables. While the requirements on the design variables are not fully decoupled anymore, this approach enables to provide more design alternatives compared to intervals.

3 — Solution Spaces

The (complete) Solution Space is the domain in the space of design variables that involves all good designs, i.e. designs which satisfy all requirements under consideration. In the previous chapter it was shown that in literature many approaches exist that represent the set of good designs by intervals. In this chapter fundamentals of Solution Spaces and definitions are provided. Finally, analytic test problems, which are used in the subsequent chapters to analyze and assess the developed approaches, are proposed.

3.1 Low-dimensional representation of high-dimensional Solution Spaces

From a mathematical perspective, the Solution Space¹ is described by a set of inequalities, i.e. requirements on e.g. performance measures of a system. In presence of high-dimensional technical problems, the identification of the domain of good designs is challenging, however, an easily understandable description is beneficial in industrial development processes. The question is how to represent the set of good designs such that designs can be identified as good or bad in an easily understandable manner, such that the set is represented as completely as possible and such that it is in accordance with design methodologies, such as concurrent engineering. Figure 3.1 shows two possibilities to represent high-dimensional Solution Spaces: intervals and so called 2d-spaces. For intervals, every design with all design variable values within their associated intervals is good. For 2d-spaces, every design with all design variable values within their associated 2d-spaces (white areas in Figure 3.1, bottom right) is good.

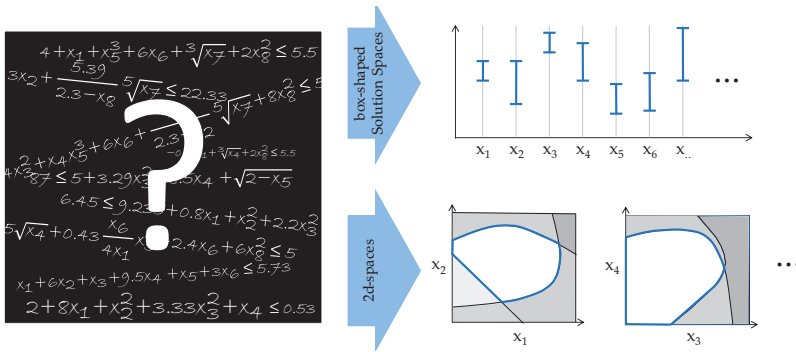


Figure 3.1: From a mathematical perspective, the Solution Space is described by a set of inequalities. Consider the left side of the figure: Can you find the combinations of design variable values, i.e. values of x_1, x_2, \dots , which lead to a good design? A low-dimensional representation of the Solution Space shows good designs in an easy understandable manner, e.g. by intervals (top right) or by 2d-spaces (white areas, bottom right).

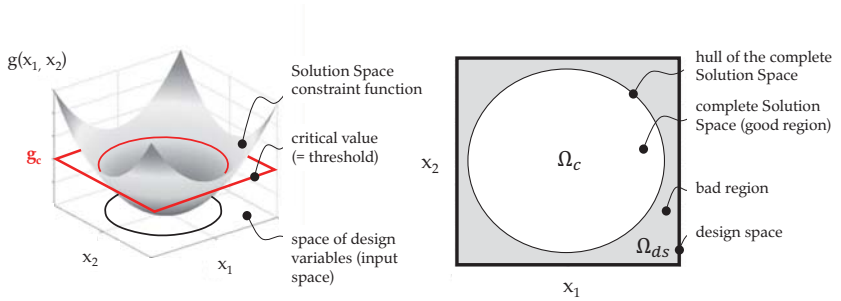


Figure 3.2: Here the set of good designs is specified by one single Solution Space constraint function (sphere function: $g(\mathbf{x}) = x_1^2 + x_2^2$), which depends on two design variables x_1 and x_2 as well as the critical value g_c as upper threshold. This excludes designs in the input space, the remaining domain is the complete Solution Space Ω_c (right).

3.2 Definitions

3.2.1 The (complete) Solution Space

Definition 1. The permissible domain, here called **complete Solution Space**² Ω_c , is the set of designs satisfying the **Solution Space constraints** $g(\mathbf{x}) \leq g_c$:

$$\Omega_c = \{\mathbf{x} \in \Omega_{ds} \mid g(\mathbf{x}) \leq g_c\} \quad (3.1)$$

with the function values, written as a vector $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^m$ and $\mathbf{g}_c \in \mathbb{R}^m$ as the vector of the associated **thresholds**. m is the **number of Solution Space constraints**. A **design** is represented by a vector \mathbf{x} containing the **design variables**, i.e. $\mathbf{x} = (x_1, x_2, \dots, x_d)^T \in \mathbb{R}^d$ with d as the **number of dimensions**. Each design variable is constrained by a **lower and an upper bound**, i.e. $x_i^{lb} \leq x_i \leq x_i^{ub}$, $i = 1, \dots, d$ with $\Omega_{ds} = [x_1^{lb}, x_1^{ub}] \times [x_2^{lb}, x_2^{ub}] \times \dots \times [x_d^{lb}, x_d^{ub}]$ as the **design space**. A design within the design space is called **good** if all the requirements $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$ are satisfied component-by-component, otherwise it is called **bad**. In this thesis, any subset $\Omega \subset \Omega_c$ of the complete Solution Space is called **Incomplete Solution Space** or a **Solution Space**. They are, e.g., obtained by considering additional conditions such as partial or complete independency of variable selection. For example, **box-shaped Solution Spaces** or **2d-spaces** are such subsets of Ω_c .

¹Note that the term *space* might be misleading referring to the mathematical definition of a space, however, here the term Solution Space is used in accordance with the literature.

²In the field of numerical optimization, the complete Solution Space is also called feasible space.

Figure 3.4 shows different types of Solution Space constraints and the associated shape of the complete Solution Space. Figure 4.6 shows an example, where the box-shaped Solution Space is a subset of the complete Solution Space.

3.2.2 Parametric vs. oracle functions and surrogates

In the subsequent sections, two different types of functions, namely a **parametric function** (also called **white-box**) and an **oracle function** (also called **black-box**) are considered. The terms and the following definitions are in accordance with Boyd [13]. A *parametric function* is given in analytic and closed form, i.e. can be expressed as a formula or expression, with the design variables x_i as the independent variables. In addition to the variables, it involves some parameters, e.g. the coefficients of a linear equation. An *oracle function* is characterized by the fact that nothing is known about the structure of the function. In particular, solving the equations of motion of complex dynamic systems is often not possible in closed form, i.e. the response of the system cannot be expressed as a function of the design variables, but can only be obtained by numerical integration. Hence, the response of the system can often only be obtained by evaluating an oracle function.

In practice, however, it is often essential to know the structure of the problem. The approaches proposed in this thesis make advantage of particular properties of functions and therefore, a parametric function is preferred. Two common approaches to obtain parametric functions are: Firstly, physical models are simplified, such that they can be solved in closed form. Secondly, a design of experiment, see e.g. [5, 49], on the oracle function is performed and parametric functions are derived by applying e.g. parametric regression analysis, linear support vector machine, etc. Therefore, designs are generated within a predefined design space (= sampling), are evaluated and one of the aforementioned approaches is applied. The number of sample points, which are necessary to train a reliable model, increases significantly with the number of (sensitive/relevant) design variables. Other approaches, such as neural networks or nonlinear support vector machines, generally yield a black-box function. Although, black-box models provide no insight into the mathematical structure of the system, they are able to reduce the time for the evaluation of f to obtain the system's response z significantly. The result of these techniques is an approximation of f , also known as **mathematical surrogate model**. The quality of the approximation is assessed by particular measures, e.g. the correlation between the output of the obtained approximation and the true system's response. Other measures, particularly for binary classification, are the fraction of sample points which are of category A and classified as category A (= true positive), the fraction of sample points which are of category B and classified as category B (= true negative), the fraction of sample points which are of category B but are classified as category A (= false positive) and vice

versa (= false negative). These four measures are often visualized in a confusion matrix; see [57]. An example is provided in Appendix B.1. For surrogate models in the field of machine learning; see e.g. [47].

3.2.3 The hull of a Solution Space

Normally, the complete Solution Space is surrounded by a **hull** (or **boundary**) that separates good from bad designs³. This hull is expressed as a function of the design variables in the $(d - 1)$ -dimensional subspace of the d -dimensional design space. In case of one single performance function, the mathematical description of the hull can be obtained by solving the equation $f(\mathbf{x}) = z_c$ for one of the design variables x_i . However, in general, the hull cannot be described mathematically in closed form. A common approach is to approximate the hull by applying techniques in the field of design of experiments, see [5, 49], in conjunction with *classifiers*, e.g. support vector machine [14]. Therefore, designs are generated within a pre-defined design space (= sampling), are evaluated, categorized (labeled) as good or bad w.r.t. certain criteria, and a classification approach is applied. In dependence on the classifier approach, the hull is described by a black-box or a white-box function. The number of sample points, necessary to train a reliable classifier, increases significantly with the number of (sensitive/relevant) design variables. In [9] an adaptive sampling scheme for generating reliable classifiers with relatively low effort, i.e. number of sample points and hence function evaluations is introduced. In [43] different approaches to describe a Solution Space (here it is called design space) such as convex hull method [17], prediction error variance, support vector machine and support vector machine with a leave-one-out optimization are analyzed and applied to experimental data. The quality of such models can be assessed by a confusion matrix; see Section 3.2.2. As an illustrative example, Figure 3.3 left shows the hull of the Solution Space derived by solving the equation $x_1^2 + x_2^2 = z_c$ for x_1 and Figure 3.3 right shows the result of applying support vector machine to a set of labeled data.

3.2.4 Size measure of a Solution Space

As defined in Equation (3.1), a Solution Space is a set of good designs. In mathematics, the size of a set, containing a finite number of elements (or objects) is quantified by the number of elements. For a set of infinite elements, however, this is not possible and in mathematics the cardinal number (in German also "Mächtigkeit"), defined by Georg Ferdinand Ludwig Philipp Cantor, the inventor of set theory, was introduced. See [20, 65]. For instance, the set of natural numbers \mathbb{N} and real numbers \mathbb{R} is infinite, while a subset of \mathbb{N} , e.g. $\{1, 2, 3\}$, with a cardinal number of three, is finite.

³The Solution Space is assumed to be connected.

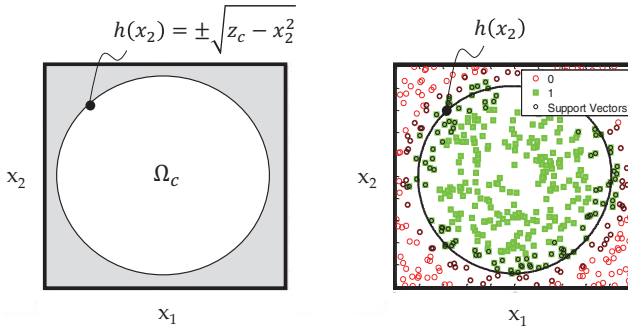


Figure 3.3: The mathematical description of the hull of a Solution Space (the Solution Space constraint is given by $x_1^2 + x_2^2 \leq z_c$) is obtained by solving the equation $x_1^2 + x_2^2 = z_c$ for x_1 (left) and as a result of sampling in conjunction with a classification approach (here support vector machine) (right).

Definition 2. In the context of Solution Spaces, the *size measure* of the set of good designs is denoted by $\mu(\Omega)$, where Ω contains, in the case of continuous design variable values $\mathbf{x} \in \mathbb{R}^d$, an infinite number of designs. $\mu(\Omega)$ is assumed as the volume in a geometrical sense:

$$\mu(\Omega) = \int_{\Omega} d\Omega. \quad (3.2)$$

Definition 2 is motivated by the fact that the volume correlates to the number of enclosed designs, which in turn correlates to robustness and flexibility in development processes.

3.2.5 Loss of Solution Space

The shape of the Solution Space depends on the Solution Space constraints. In order to visualize a high-dimensional Solution Space with arbitrary shape, it is necessary to approximate the Solution Space, e.g. by a Cartesian product of intervals (= box) and by a Cartesian product of 2d-spaces respectively. However, in general these approximations are, due to geometrical mismatch, subsets of the complete Solution Spaces only; see e.g. Figure 4.6 for box-shaped Solution Spaces or Figure 5.1 for box-shaped Solution Spaces and 2d-spaces.

Definition 3. Assume the size of a set of designs given by a size measure $\mu(\Omega)$. Then the *loss of Solution Space* can be quantified as the ratio of the size measure of the whole






	type of Solution Space constraint	shape of complete Solution Space	example	
A	linear	convex, connected	Tilted-Hyperplane	
B	nonlinear, monotone , convex	convex, connected	Sphere	
C	nonlinear, monotone , non-convex	non-convex, connected	(-1) Sphere	
D	non-monotone , convex	convex, connected	Sphere	
E	non-monotone , non-convex	non-convex, connected or non-connected	(-1) Sphere	

Figure 3.4: Relation between the type of Solution Space constraint and the associated shape of the complete Solution Space.

set of good designs, i.e. complete Solution Space $\mu(\Omega_c)$, to the size measure of a selected subset $\mu(\Omega)$, e.g. represented by the Cartesian product of intervals or by the Cartesian product of 2d-spaces:

$$\psi = \frac{\mu(\Omega_c)}{\mu(\Omega)}. \quad (3.3)$$

Note that loss of Solution Space does not only occur in case of a representation of the complete Solution Space by boxes or 2d-spaces but also by approximating the true complete Solution Space by mathematical surrogates, see Section 3.2.2.

3.2.6 Linearity, monotonicity, convexity

Most of the problem statements in Chapter 4 and 5 require particular types of Solution Space constraints and make advantage of the associated properties in terms of simplifications and consequently efficiency. The following mathematical statements are written for a continuous function $g : \mathbb{R}^d \mapsto \mathbb{R}$.

Definition 4. The function g is linear w.r.t. the i -th design variable, if it satisfies the

following condition:

$$\begin{aligned} g(\mathbf{x}_{\sim i}) - g(\mathbf{x}) &= g(\tilde{\mathbf{x}}_{\sim i}) - g(\tilde{\mathbf{x}}) \\ \forall \mathbf{x}, \mathbf{x}_{\sim i}, \tilde{\mathbf{x}}, \tilde{\mathbf{x}}_{\sim i} &\in \Omega_{ds}. \end{aligned} \quad (3.4)$$

With $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ as an arbitrary design, where the i -th component is shifted by δx_i . \mathbf{e}_i is the unit vector with entry in the i -th component. $\tilde{\mathbf{x}}$ is any other design than \mathbf{x} .

Equation (3.4) states that the derivative w.r.t. a certain design variable is equal for all \mathbf{x} within the design space.

Remark. Boyd [13] mentions the following condition for linear functions:

$$\begin{aligned} g(\alpha \mathbf{x} + \beta \tilde{\mathbf{x}}) &= \alpha g(\mathbf{x}) + \beta g(\tilde{\mathbf{x}}) \\ \forall \mathbf{x}, \tilde{\mathbf{x}} &\in \Omega_{ds} \end{aligned} \quad (3.5)$$

for all $\alpha, \beta \in \mathbb{R}$.

Definition 5. The function g is monotonically increasing w.r.t. the design variable x_i if it satisfies the following condition:

$$\begin{aligned} g(\mathbf{x}_{\sim i}) &\geq g(\mathbf{x}) \\ \forall \mathbf{x}, \mathbf{x}_{\sim i} &\in \Omega_{ds}, \delta x_i > 0 \end{aligned} \quad (3.6)$$

with $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ as an arbitrary design, where the i -th component is shifted by δx_i . \mathbf{e}_i is the unit vector with entry in the i -th component. A function g is monotonically decreasing w.r.t. the design variable x_i if it satisfies the following condition:

$$\begin{aligned} g(\mathbf{x}_{\sim i}) &\leq g(\mathbf{x}) \\ \forall \mathbf{x}, \mathbf{x}_{\sim i} &\in \Omega_{ds}, \delta x_i > 0. \end{aligned} \quad (3.7)$$

Definition 5 is in accordance with [31].

Definition 6. The function g is convex if it satisfies the following inequality (in accordance with [13]):

$$\begin{aligned} g(\alpha \mathbf{x} + (1 - \alpha) \tilde{\mathbf{x}}) &\leq \alpha g(\mathbf{x}) + (1 - \alpha) g(\tilde{\mathbf{x}}) \\ \forall \mathbf{x}, \tilde{\mathbf{x}} &\in \Omega_{ds}, \mathbf{x} \neq \tilde{\mathbf{x}} \end{aligned} \quad (3.8)$$

for all $\alpha \in \mathbb{R}$ with $\alpha \in [0, 1]$ and with $\tilde{\mathbf{x}}$ being any other design than \mathbf{x} .

Definition 7. The function g is convex w.r.t. the design variable x_i if it satisfies the following inequality:

$$\begin{aligned} g(\alpha \mathbf{x} + (1 - \alpha) \mathbf{x}_{\sim i}) &\leq \alpha g(\mathbf{x}) + (1 - \alpha) g(\mathbf{x}_{\sim i}) \\ \forall \mathbf{x}, \mathbf{x}_{\sim i} \in \Omega_{ds}, \delta x_i &> 0 \end{aligned} \quad (3.9)$$

for all $\alpha \in \mathbb{R}$ with $\alpha \in [0, 1]$ and with $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ as an arbitrary design, where the i -th component is shifted by δx_i . \mathbf{e}_i is the unit vector with entry in the i -th component.

Definition 8. Let the function g be twice differentiable and let \mathbf{H} be the Hessian of g , a symmetric matrix. The function is convex if the eigenvalues, i.e. the values of λ as solution of the equation

$$\det(\mathbf{H} - \lambda \mathbf{I}) = 0 \quad (3.10)$$

are greater or equal than zero. \mathbf{I} is the identity matrix.

3.3 Deriving Solution Space constraints from performance functions

One possibility to obtain the set of inequalities that describes the complete Solution Space, is to derive \mathbf{g} and \mathbf{g}_c directly from a set of m requirements on performance measures $\mathbf{z} = \mathbf{f}(\mathbf{x})$. $\mathbf{z} \in \mathbb{R}^m$ is the **vector of performance measures** and \mathbf{f} is the **vector of performance functions**, each mapping the design variables $\mathbf{x} \in \mathbb{R}^d$ to \mathbf{z} . The requirements are given either by lower and upper or by **lower or upper bounds on the performance measures** z_c , and hence the *Solution Space constraints* in standard form $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$ can be derived. For example, for a requirement of the form $f(x) \leq z_c$, the Solution Space constraint function is $g = f$ and the associated threshold is $g_c = z_c$. For a requirement of the form $f(x) \geq z_c$, the Solution Space constraint function is $g = -f$ and the threshold is $g_c = -z_c$.

Alternatively, either the performance functions or the hull of the complete Solution Space can be approximated by applying techniques such as linear parametric regression, see [33], or machine learning; see [47]. The result of these methods are mathematical surrogates (also called response surface or meta-model); see Section 3.2.2. The motivation is often a lower computational cost for evaluating the constraints or a desired insight into the structure of the model.

In summary, the Solution Space constraints are derived:

- directly from the performance functions.
- from an approximation of the performance functions.

- from a mathematical description of the hull of the Solution Space, e.g. obtained by applying classifiers such as support vector machine or convex hull algorithms [17].

3.4 Assessing sets regarding Solution Space constraints

For various reasons, e.g. better visualization of a high-dimensional Solution Space, often subsets of the complete Solution Space are sought, e.g. box-shaped Solution Spaces or 2d-spaces. Ω is a subset of the complete Solution Space Ω_c as long as it contains good designs only. However, the assessment of the Solution Space constraints *for all \mathbf{x} within the set* for subsets with arbitrary shape is computationally very expensive, if not impossible. If the constraints in (3.1) can be assessed by a finite number of function evaluations it is called **direct evaluation**, otherwise **indirect evaluation**.

In the case of *discrete design variable values*, a direct evaluation is possible and the number of designs to be evaluated depends on the number of elements of the subset; see Section 3.2.4.

In the case of *continuous design variable values*, e.g. if $\mathbf{x} \in \mathbb{R}^d$, the Solution Space contains an infinite number of designs, and hence a direct evaluation of a subset *with arbitrary shape* is generally impossible.

However, if the subset is specified by a box or, more generally, by a polytope, it might be sufficient to assess only the vertexes or particular vertexes of the polytope to ensure that the enclosed set contains good designs only. In this thesis this is called **vertex tracking** and a direct evaluation is possible even for cases where \mathbf{x} is continuous. The validity of this approach depends on the type of Solution Space constraints and is discussed particularly for boxes in Chapter 4 and for polytopes in Chapter 5.

If a direct evaluation is impossible the satisfaction of the constraints can be assessed by sampling in conjunction with statistical statements; see also Section 4.2.2.

3.5 Analytic test problems

In the following, analytic test problems, i.e. parametric functions, are provided to evaluate the proposed algorithms in the subsequent chapters. Multiple linear and nonlinear problems have been selected, such that a wide range of types of real-world problems is represented. The types are:

- Linear problems, i.e. the Solution Space constraints are linear functions (Tilted-Hyperplane, Four-Tilted-Hyperplanes);
- Nonlinear monotone convex and non-convex problems, i.e. the Solution Space constraints are monotone convex and non-convex functions w.r.t. the design variables

(Sphere);

- Nonlinear non-monotone convex and non-convex problems, i.e. the Solution Space constraints are non-monotone but convex and non-convex functions w.r.t. the design variables (Sphere);

An overview as well as additional information about the type of Solution Space constraint, the name of the function used, the design space as well as the initial box for the numerical evaluations are provided in Table 3.1. The shape of the Solution Spaces in two dimensions is shown in Figure 3.5. The formula of the analytic solution of the optimal size measure $\mu(\Omega)$ for each test problem and for each approach, listed in Figure 1.10, is provided in Table 3.2. The formulas are derived by analytic calculus. For the derivation for the size measure of an optimal box for test problem *A1* see [36], for the derivation for the size measure of an optimal decomposition into 2d-spaces for test problem *A1* see [26]. In approach *W1*, see Section 5.4, the Solution Space is represented by orthogonal polygons with a specified number of vertexes, denoted by p . Here, the formula for the case where the two-dimensional polygons have four and six vertexes, respectively is provided. Additionally, the formula for the optimal size measure of the outer box, see Section 4.2.1, is shown.

Table 3.1: Categorization of the test problems considered as well as the associated design spaces and initial boxes for the numerical computation of optimal Solution Spaces.

	type of Solution Space constraint	function name	design space	initial box
<i>A1</i>	linear	Tilted-Hyperplane	$[0, 1]^d$	$[0.01, 0.11]^d$
<i>A2</i>	linear	Four-Tilted-Hyperplanes	$[0, 1]^d$	$[0.45, 0.55]^d$
<i>B1</i>	nonlinear, monotone, convex	Sphere	$[0, 1]^d$	$[0.01, 0.11]^d$
<i>C1</i>	nonlinear, monotone, non-convex	Sphere	$[0, 1]^d$	$[0.01, 0.11]^d$
<i>D1</i>	non-monotone, convex	Sphere	$[0, 1]^d$	$[0.45, 0.55]^d$
<i>E1</i>	non-monotone, non-convex	Sphere	$[0, 1]^d$	$[0.01, 0.11]^d$

Tilted-Hyperplane The Tilted-Hyperplane problem (*A1*) is specified by the following inequality

$$\sum_{i=1}^d x_i \leq \frac{d}{2} \tag{3.11}$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, d.$$

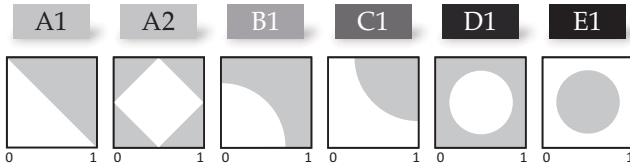


Figure 3.5: Overview of the test problems considered.

Four-Tilted-Hyperplanes The Tilted-Hyperplane problem is modified such that it comprises four constraints (*A2*):

$$\begin{aligned}
 \sum_{i=1}^d x_i &\leq \frac{3}{4}d \\
 -\sum_{i=1}^d x_i &\leq -\frac{1}{4}d \\
 -\sum_{i=1}^{d/2} x_{2i-1} + \sum_{i=1}^{d/2} x_{2i} &\leq \frac{1}{4}d \\
 \sum_{i=1}^{d/2} x_{2i-1} - \sum_{i=1}^{d/2} x_{2i} &\leq \frac{1}{4}d \\
 0 \leq x_i \leq 1, \quad i = 1, \dots, d.
 \end{aligned} \tag{3.12}$$

Note that the Four-Tilted-Hyperplanes problem is valid for an even number of dimensions only.

Sphere The Sphere problem (*B1*) is specified by the following inequality

$$\begin{aligned}
 \sum_{i=1}^d x_i^2 &\leq \frac{d}{4} \\
 0 \leq x_i \leq 1, \quad i = 1, \dots, d.
 \end{aligned} \tag{3.13}$$

For (*C1*) the problem reads

$$\begin{aligned}
 -\sum_{i=1}^d x_i^2 + 2 \sum_{i=1}^d x_i &\leq \frac{3}{4}d \\
 0 \leq x_i \leq 1, \quad i = 1, \dots, d.
 \end{aligned} \tag{3.14}$$

For (*D1*) the problem reads

$$\sum_{i=1}^d x_i^2 - \sum_{i=1}^d x_i \leq -\frac{3}{16}d \quad (3.15)$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, d.$$

For (*E1*) the problem reads

$$-\sum_{i=1}^d x_i^2 + \sum_{i=1}^d x_i \leq \frac{3}{16}d \quad (3.16)$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, d.$$

Table 3.2: Analytic solutions of the optimal, i.e. maximum, size measures for the test problems *A1*, *A2*, *B1*, *C1*, *D1* and *E1*. The Solution Space is represented by intervals (approach *U1* and *V1*) and 2d-spaces (approach *W1* and *W2*), respectively. For approach *W1* the Solution Space is represented by orthogonal two-dimensional polygons with four and six vertexes, respectively. Additionally, the size measure of the optimal outer box is provided. Note that the number of dimensions *d* is assumed to be even.

	<i>A1</i>	<i>A2</i>	<i>B1</i>	<i>C1</i>	<i>D1</i>	<i>E1</i>
<i>U1, V1</i>	0.5^d	0.5^d	0.5^d	0.5^d	0.5^d	-
<i>W1, p = 4</i>	$0.5^{d/2}$	$0.5^{d/2}$	$\left(\frac{\sqrt{2}}{4}\right)^{d/2}$	-	$\left(\frac{1}{4}\right)^{d/2}$	-
<i>W1, p = 6</i>	$0.5^{d/2}$	$0.5^{d/2}$	$\left(\frac{\sqrt{2-\sqrt{2}}}{2}\right)^{d/2}$	-	$\left(\frac{3\sqrt{3}}{16}\right)^{d/2}$	-
<i>W2</i>	$0.5^{d/2}$	$0.5^{d/2}$	$\left(\frac{\pi}{8}\right)^{d/2}$	$\left(1 - \frac{\pi}{8}\right)^{d/2}$	$\left(\frac{\pi}{8}\right)^{d/2}$	$\left(1 - \frac{\pi}{8}\right)^{d/2}$
<i>outer box</i>	1	1	$\left(\frac{\sqrt{d}}{2}\right)^d$ *	1	$\left(\frac{\sqrt{d}}{2}\right)^d$ *	-

* $\in [0, 1]$

4 — Box-shaped Solution Spaces

Box-shaped Solution Spaces represent a high-dimensional Solution Space by intervals for each design variable. From a geometrical perspective, each interval is an edge of the box. In order to enclose as many good designs as possible, the volume of the box must be maximized. In this chapter mathematical formulations of the underlying optimization problem are stated, a stochastic approach to compute box-shaped Solution Spaces is reviewed, an approach in which the box is assessed by tracking the vertexes of the box is proposed and numerical results based on the analytic test problems as well as modifications of the introduced problem statements are presented.

4.1 Intervals as representation of high-dimensional Solution Spaces

One possibility to represent a high-dimensional Solution Space with arbitrary shape is to seek a box within the design space that contains good designs only. Each edge of the box is an interval associated with a particular design variable that provides an intuitive description of the high-dimensional problem: If each design variable value of a design is within its associated interval, all the system requirements are satisfied. From a practical perspective, two main advantages are:

- Decision makers get an overview of design alternatives easily.
- If there is a decision maker for each design variable separately, the decisions can be made independently from each other.

Further advantages of box-shaped Solution Spaces are mentioned in Section 2.3. In order to represent the entire Solution Space by the box as completely as possible, the size of the box must be maximized.

4.2 General problem statement

The problem of seeking a *box with maximum box size measure* $\mu(\Omega)$ that contains good designs only is written as follows (in accordance with [78]):

$$\begin{aligned} & \underset{I_1, I_2, \dots, I_d}{\text{maximize}} && \mu(\Omega) \\ & \text{s.t.} && \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c \quad \forall \mathbf{x} \in \Omega \subseteq \Omega_{\text{ds}} \end{aligned} \quad (4.1)$$

with the *box size measure* $\mu(\Omega)$ defined as

$$\mu(\Omega) = \prod_{i=1}^d \mu(I_i) = \prod_{i=1}^d (x_i^{\text{u}} - x_i^{\text{l}}). \quad (4.2)$$

The **box** Ω is the Cartesian product of intervals denoted by I_i , i.e. $\Omega = I_1 \times I_2 \times \dots \times I_d = [x_1^{\text{l}}, x_1^{\text{u}}] \times [x_2^{\text{l}}, x_2^{\text{u}}] \times \dots \times [x_d^{\text{l}}, x_d^{\text{u}}]$. x_i^{l} and x_i^{u} denote the **lower and upper boundary of an interval**, associated with the i -th design variable. The box size measure is the volume of the box as motivated in Section 3.2.4. The constraints of the optimization problem (4.1) are equivalent to (3.1), i.e. all *Solution Space constraints* must be satisfied by all designs within the box.

4.2.1 Concept of inner and outer box

Definition 9. *The inner box is defined as a box, that contains good designs only and hence lies within the hull of the complete Solution Space. Therefore, the inner box is a subset of the complete Solution Space; see also Section 3.2.1:*

$$\begin{aligned} \Omega_{in} &= [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \dots \times [x_d^l, x_d^u] \subseteq \Omega_c \\ \text{with } \mathbf{g}(\mathbf{x}) &\leq \mathbf{g}_c \quad \forall \mathbf{x} \in \Omega_{in}. \end{aligned} \quad (4.3)$$

The statement for an inner box is: Designs within that box are strictly good and designs outside of that box may be good.

Definition 10. *The outer box, see also [62], is defined as a box, where all designs outside that box are bad:*

$$\begin{aligned} \Omega_{out} &= [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \dots \times [x_d^l, x_d^u] \subseteq \Omega_{ds} \\ \text{with } \exists j \in \{1, \dots, m\} : g_j(\mathbf{x}) &> g_{c,j} \quad \forall \mathbf{x} \notin \Omega_{out}. \end{aligned} \quad (4.4)$$

The statement for an outer box is: Designs outside of that box are strictly bad and designs within that box may be bad.

On the left side of Figure 4.1, an inner box, satisfying condition (4.3) as well as an outer box satisfying condition (4.4) are shown. On the right side, the maximum inner box as well as the minimum outer box, i.e. with maximum and minimum box size measure, are shown.

While the statement about an inner box is strong in terms of feasibility and *weak in terms of infeasibility* (one may find more solutions outside that box), the statement about the outer box is strong in terms of infeasibility (one does not find any solution outside that box). Often the *minimum outer box* is of interest, since the design space can be decreased to that box and hence the space for searching for a good design or an inner box decreases, and consequently the computational effort. The problem statement for seeking the *minimum outer box* reads

$$\begin{aligned} \text{minimize } & \mu(\Omega) \\ \text{s.t. } & \exists j \in \{1, \dots, m\} : g_j(\mathbf{x}) > g_{c,j} \quad \forall \mathbf{x} \in \Omega_{ds} \setminus \Omega. \end{aligned} \quad (4.5)$$

The constraint of the optimization problem is satisfied, if all designs outside the box fail w.r.t. one or more Solution Space constraints, i.e. if all designs outside the box are bad.

4.2.2 Assessing box-shaped sets regarding Solution Space constraints

The constraints of optimization problem (4.1) and the constraints involved in the definitions of the inner and outer box (4.3) and (4.4), namely $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$, for a set of designs

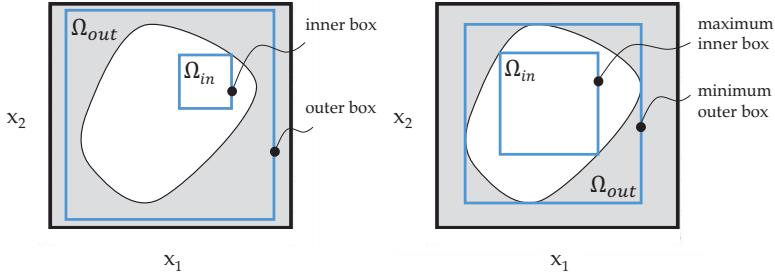


Figure 4.1: An inner and an outer box (left) as well as the maximum inner box and the minimum outer box (right).

can be evaluated either in a *direct* or an *indirect* manner; see Section 3.4.

For a direct evaluation in the case of continuous design variable values, the approach of *vertex tracking* was introduced in Section 3.4. Note that tracking vertexes of a box is related to interval arithmetic [3] or interval propagation in the framework of set-based design [31, 32], i.e. only (particular) bounds of input intervals (= vertexes of a hyperbox) are considered to determine the range of an output. In the framework of box-shaped Solution Spaces, the minimum and maximum values of an output for all designs within the box is of interest. The computation with intervals rather than real numbers is based on a set of axioms. In the following, the basic operations of interval arithmetic for *addition*, *subtraction*, *multiplication* and *division* for two real compact intervals $I_1 = [x_1^l, x_1^u]$ and $I_2 = [x_2^l, x_2^u]$ are provided (in accordance with [62]).

$$\begin{aligned}
 I_1 + I_2 &= [x_1^l + x_2^l, x_1^u + x_2^u] \\
 I_1 - I_2 &= [x_1^l - x_2^u, x_1^u - x_2^l] \\
 I_1 I_2 &= [\min(x_1^l x_2^l, x_1^l x_2^u, x_1^u x_2^l, x_1^u x_2^u), \max(x_1^l x_2^l, x_1^l x_2^u, x_1^u x_2^l, x_1^u x_2^u)] \\
 \frac{I_1}{I_2} &= [x_1^l, x_1^u] \left[\frac{1}{x_2^u}, \frac{1}{x_2^l} \right], 0 \notin I_2.
 \end{aligned} \tag{4.6}$$

However, if a variable occurs more than once in the output function (e.g. $f(x) = x(1-x)^{-1}$ with $x = [2, 3]$, example taken from [3]), the output range may be overestimated (in the example mentioned above $[-3, -1]$ instead of $[-2, -1.5]$). In the literature, many approaches are proposed to reduce or avoid overestimation, see e.g. [3], particularly for monotone functions see [31, 32]. Note that applying interval arithmetic requires the functions to be parametric functions.

In order to provide a mathematical condition for the validity of the vertex tracking approach for box-shaped Solution Spaces, the **one-dimensional sublevel set**, see also

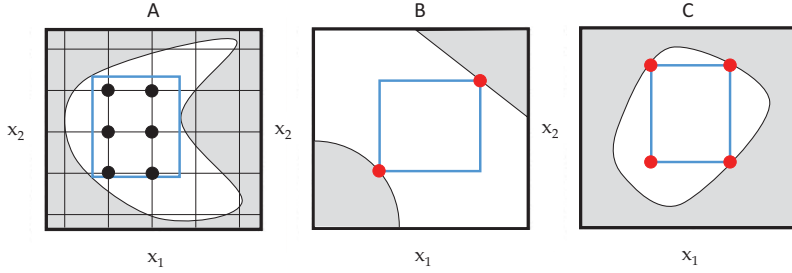


Figure 4.2: Direct evaluation of a box-shaped Solution Space by (A) assessing the finite number of designs inside the box (in case of discrete design variable values), (B) assessing particular vertexes of the box (in case of monotone Solution Space constraints) and (C) assessing all vertexes of the box (in case of convex Solution Spaces).

[13], is introduced. The sublevel set of the i -th design variable is the one-dimensional set of good designs parallel to the associated axis of the design space, i.e. a section of the high-dimensional Solution Space; see Figure 4.5.

Definition 11. The sublevel set $\mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x}))$ of the i -th design variable x_i for given Solution Space constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$ and a design \mathbf{x} is defined as

$$\mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x})) = \{x_i \in [x_i^{lb}, x_i^{ub}] \mid \mathbf{g}(\mathbf{x}_{\sim i}) \leq \mathbf{g}_c\} \quad (4.7)$$

$i = 1, \dots, d.$

With $\mathbf{x}_{\sim i} = \mathbf{x} + (x_i - e_i^T \mathbf{x}) \mathbf{e}_i$ as an arbitrary design, where the i -th component is replaced by x_i . \mathbf{e}_i is the unit vector with entry in the i -th component.

Theorem 1. Let all sublevel sets as defined in (4.7) be convex:

$$\mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x})) \text{ is convex } \forall \mathbf{x} \in \Omega_c \quad (4.8)$$

$\forall i = 1, \dots, d.$

Then, in order to avoid regions of bad designs within a box, it is sufficient to track the vertexes of the box.

Proof. Let $\Omega = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \dots \times [x_d^l, x_d^u]$ be a hyperbox and let $C_0 = \{x_1^l, x_1^u\} \times \{x_2^l, x_2^u\} \times \dots \times \{x_d^l, x_d^u\}$ be the vertexes of the box satisfying the Solution Space constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$. Furthermore, let all sublevel sets $\mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x}))$, as defined in (4.7), be convex.

It must be shown that for arbitrary $\tilde{\mathbf{x}} \in \Omega$ with $\tilde{x}_i = \alpha_i x_i^l + (1 - \alpha_i) x_i^u$, $\alpha_i \in [0, 1]$, $i = 1, \dots, d$ it holds $\mathbf{g}(\tilde{\mathbf{x}}) \leq \mathbf{g}_c$.

For $\mathbf{x} \in C_0$, it is $\mathbf{x}_{\sim 1}^l = \mathbf{x} + (x_1^l - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1 \in C_0$, $\mathbf{x}_{\sim 1}^u = \mathbf{x} + (x_1^u - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1 \in C_0$ and $x_1^l, x_1^u \in \mathcal{S}_{g_{c,1}}(\mathbf{g}(\mathbf{x}))$. As $\tilde{x}_1 = \alpha_1 x_1^l + (1 - \alpha_1) x_1^u$, $\alpha_1 \in [0, 1]$ we get $\tilde{x}_{\sim 1} \in \mathcal{S}_{g_{c,1}}(\mathbf{g}(\mathbf{x}))$ and therefore, the inequality $\mathbf{g}(\tilde{\mathbf{x}}_{\sim 1}) \leq \mathbf{g}_c$ is valid for $\tilde{\mathbf{x}}_{\sim 1} = \mathbf{x} + (\tilde{x}_1 - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1$. Thus, the set $C_1 = \{\tilde{\mathbf{x}}_{\sim 1} \mid \tilde{\mathbf{x}}_{\sim 1} = \mathbf{x} + (\tilde{x}_1 - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1, \mathbf{x} \in C_0\}$, where any element satisfies the Solution Space constraints, can be defined.

Recursively, for $\mathbf{x} \in C_{i-1}$, $i = 2, \dots, d$ it is $\mathbf{x}_{\sim i}^l = \mathbf{x} + (x_i^l - \mathbf{e}_i^T \mathbf{x}) \mathbf{e}_i \in C_{i-1}$, $\mathbf{x}_{\sim i}^u = \mathbf{x} + (x_i^u - \mathbf{e}_i^T \mathbf{x}) \mathbf{e}_i \in C_{i-1}$ and $x_i^l, x_i^u \in \mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x}))$. Also, with $\tilde{x}_{\sim i} \in \mathcal{S}_{g_{c,i}}(\mathbf{g}(\mathbf{x}))$, $\mathbf{g}(\tilde{\mathbf{x}}_{\sim i}) \leq \mathbf{g}_c$ is valid for $\tilde{\mathbf{x}}_{\sim i} = \mathbf{x} + (\tilde{x}_{\sim i} - \mathbf{e}_i^T \mathbf{x}) \mathbf{e}_i$ and the set $C_i = \{\tilde{\mathbf{x}}_{\sim i} \mid \tilde{\mathbf{x}}_{\sim i} = \mathbf{x} + (\tilde{x}_{\sim i} - \mathbf{e}_i^T \mathbf{x}) \mathbf{e}_i, \mathbf{x} \in C_{i-1}\}$ satisfying the Solution Space constraints is defined. After a total of d iterations, we get $\tilde{\mathbf{x}}$ as the only element of C_d with $\mathbf{g}(\tilde{\mathbf{x}}) \leq \mathbf{g}_c$. □

Remark. Since the intersection of convex sets yields a convex set, Equation (4.8) can be assessed for each Solution Space constraint function g_j , $j = 1, \dots, m$ separately. However, consider that the intersection of non-convex sets may yield a convex set. This implies that even if condition (4.8) fails w.r.t. one or more Solution Space constraint functions it may be satisfied, if all constraint functions are considered at once. Hence, condition (4.8) assessed for each constraint function separately is sufficient but not necessary, while it is sufficient and necessary if all constraint functions are considered at once.

Theorem 2. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be a monotone function w.r.t. all design variables, i.e. equations (3.6) and (3.7) respectively are satisfied. Then $\mathcal{S}_{g_{c,i}}(g(\mathbf{x}))$ is convex for all $\mathbf{x} \in \Omega_c$ and w.r.t. all design variables. If this applies for all Solution Space constraint functions, condition (4.8) is satisfied.

Proof. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be a monotone function and let $\mathbf{x}_{\sim i}$ be a design \mathbf{x} with variation by δx_i in the i -th design variable, i.e. $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ with $\delta x_i \geq 0$ and \mathbf{e}_i as the unit vector with entry in the i -th component. With $x_i, x_i + \delta x_i \in \mathcal{S}_{g_{c,i}}(g(\mathbf{x}))$ it follows $g(\mathbf{x}) \leq g_c$ and $g(\mathbf{x}_{\sim i}) \leq g_c$. Due to the properties of monotonicity, stated in Equation (3.6) and (3.7), for monotone increasing functions it holds $g(\alpha \mathbf{x} + (1 - \alpha) \mathbf{x}_{\sim i}) \in [g(\mathbf{x}), g(\mathbf{x}_{\sim i})]$ and for monotone decreasing functions it holds $g(\alpha \mathbf{x} + (1 - \alpha) \mathbf{x}_{\sim i}) \in [g(\mathbf{x}_{\sim i}), g(\mathbf{x})]$. Hence, it follows $g(\alpha \mathbf{x} + (1 - \alpha) \mathbf{x}_{\sim i}) \leq g_c \forall \alpha \in [0, 1]$, i.e. $\alpha \mathbf{x} + (1 - \alpha) \mathbf{x}_{\sim i} \in \mathcal{S}_{g_{c,i}}(g(\mathbf{x}))$. □

Theorem 3. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function w.r.t. all design variables, i.e. Equation (3.9) is satisfied. Then $\mathcal{S}_{g_{c,i}}(g(\mathbf{x}))$ is convex for all $\mathbf{x} \in \Omega_c$ and w.r.t. all design variables. If this applies for all Solution Space constraint functions, condition (4.8) is satisfied.

Proof. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function w.r.t. all design variables and let $\mathbf{x}_{\sim i}$ be a design \mathbf{x} with variation by δx_i in the i -th design variable, i.e. $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ with $\delta x_i \geq 0$ and \mathbf{e}_i as the unit vector with entry in the i -th component. With $x_i, x_i + \delta x_i \in \mathcal{S}_{g_{c,i}}(g(\mathbf{x}))$ it follows $g(\mathbf{x}) \leq g_c$ and $g(\mathbf{x}_{\sim i}) \leq g_c$. Due to the properties of convexity, stated in Equation

(3.9), i.e. $g(\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}_{\sim i}) \leq \alpha g(\mathbf{x}) + (1 - \alpha)g(\mathbf{x}_{\sim i})$ and with $g(\mathbf{x}), g(\mathbf{x}_{\sim i}) \leq g_c$ it follows $g(\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}_{\sim i}) \leq g_c \forall \alpha \in [0, 1]$ and hence, $\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}_{\sim i} \in \mathcal{S}_{g_c, i}(g(\mathbf{x}))$. A proof is also given in [13]. \square

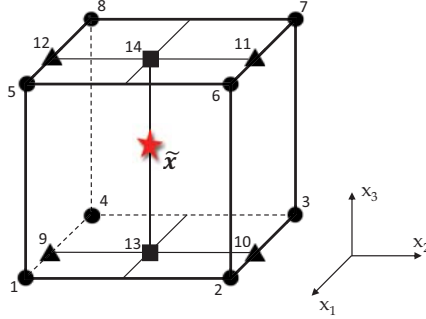


Figure 4.3: Visualization of the proof for Theorem 1 for three dimensions with the set $C_0 = \{1, 2, \dots, 8\}$ (circles), $C_1 = \{9, 10, 11, 12\}$ (triangles), $C_2 = \{13, 14\}$ (squares) and $C_3 = \{\tilde{\mathbf{x}}\}$ (star).

Figure 4.5 shows sublevel sets for different types of Solution Space constraint functions. Furthermore, the obtained interval of the i -th design variable by applying vertex tracking is shown. For vertex tracking one bound or both bounds of the interval of the i -th design variable is/are assessed only, to ensure that all designs in between are good. For monotone functions (*Type A*, *Type B* and *Type C*, see Figure 3.4) only the lower or upper boundary of each design variable must be assessed (top left). For non-monotone but convex functions (*Type D*) the lower and upper boundaries must be assessed (top right). Hence, for monotone and non-monotone but convex functions, it is sufficient to track the vertexes of the box to ensure that the box contains good designs only; see also Theorem 2 and Theorem 3. For functions that are neither monotone nor convex (*Type E*), it is generally not possible to evaluate the box by vertex tracking (bottom left: The obtained interval with both bounds satisfying the Solution Space constraint contains bad designs). However, for certain cases, vertex tracking is also possible for Solution Space constraints of *Type E* (bottom right).

Figure 4.2 (A) depicts the case where the design variable values are discrete and a direct evaluation is possible. The number of designs to be evaluated depends on the number of dimensions and the number of possible design variable values for each design variable (ϕ_i) within the box. The number of designs to be evaluated is $\prod_{i=1}^d \phi_i$. Figure 4.2 (B) shows

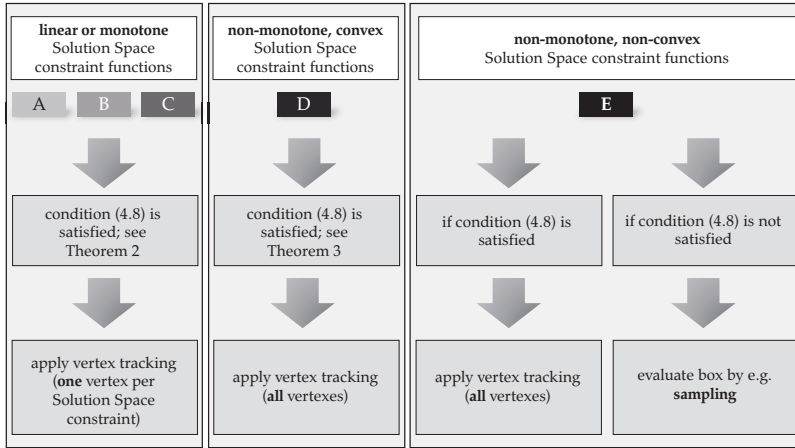


Figure 4.4: Overview of the dependencies between the types of Solution Space constraints and the validity of vertex tracking.

the case where the Solution Space constraint function g is monotone. In this case, either the lower or the upper boundary of each interval must be checked. Hence, only particular vertexes of the box must be evaluated and the number of vertexes to be assessed equals the number of constraints m . How to find these vertexes is explained in Section 4.5 and 4.6. Figure 4.2 (C) shows the case where the complete Solution Space is convex. It is sufficient to check the vertexes of the box, which are all possible combinations of lower and upper boundaries of the design variables $\{x_1^l, x_1^u\} \times \{x_2^l, x_2^u\} \times \dots \times \{x_d^l, x_d^u\}$, and hence the number of vertexes is 2^d . However, in presence of high dimensions, (A) and (C) may not be practicable due to the high number of designs to be evaluated, while in case (B) the number of evaluations depends only linearly on the number of constraints (which in industrial examples considered here typically does not exceed 100). If g contains *neither monotone nor convex* (Type E, see Figure 3.4) functions, no general statement about the validity of vertex tracking is possible and condition (4.8) must be checked for the problem on hand. In dependence on g and g_c , vertex tracking might be possible, compare Figure 4.5 bottom left and bottom right. An overview is given in Figure 4.4.

If a direct evaluation is impossible, the satisfaction of the constraints for all designs within the box can be assessed by statistics. Zimmermann et al. [78] propose the *relaxed problem statement* by reformulating the constraints of problem (4.1) as a probability statement, which is assessed by Monte Carlo sampling and Bayesian statistics; see Section 4.4.

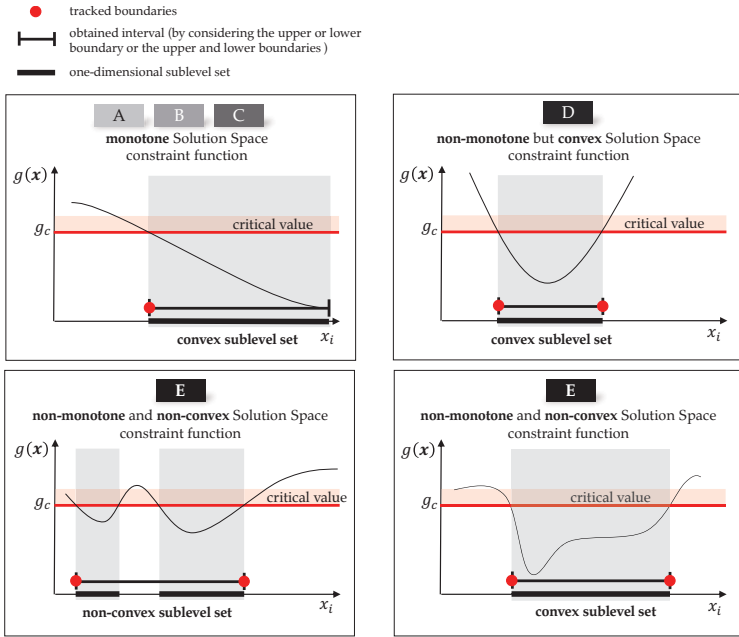


Figure 4.5: The examples show the one-dimensional sublevel sets for different types of Solution Space constraint functions g along the i -th design variable. Additionally, the interval is shown, that is obtained by applying vertex tracking, i.e. assessing one or both bounds of the interval only.

4.3 Loss of Solution Space

In the following, the chassis design problem introduced in Chapter 7 is considered with ten of the twelve design variables and vehicle parameters respectively, assuming a constant value. Hence, it is possible to visualize the complete Solution Space for two design variables, e.g. the stiffness of the anti-roll bar of the front and the rear axle. Figure 4.6 depicts the complete Solution Space as white area together with the optimal box, i.e. with maximum size measure, under consideration of the vehicle requirements listed in Table 7.2. The shaded areas are those combinations of design variable values that lead to a system that fails regarding one or more requirements. All requirements depend linearly on both of the design variables, and hence diagonal boundaries between good and bad designs within the input space occur. Figure 4.6 shows, that the optimal box is able to capture a small portion of the overall set of good designs only, and hence the loss of Solution Space

is large.

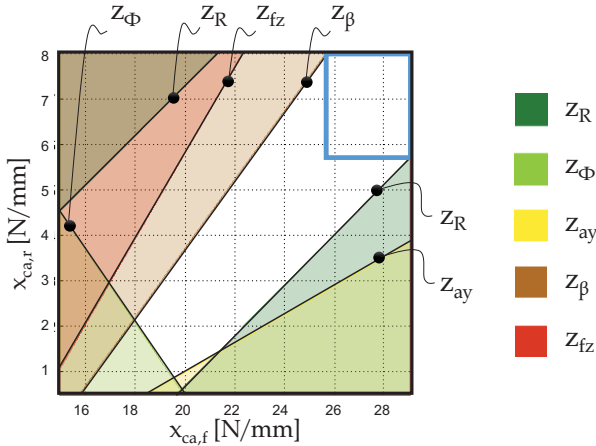


Figure 4.6: Complete Solution Space (white area) and box-shaped Solution Space (blue box) for the design variables $x_{ca,f}$ and $x_{ca,r}$ (all other variables constant).

4.4 Review: Stochastic algorithm

In this section, the algorithm presented in Zimmermann et al. [78] is briefly reviewed and the results of Graff [36], where the algorithm is analyzed in detail, are summarized.

4.4.1 Problem statement

Zimmermann et al. point out that the evaluation of the constraints of problem statement (4.1) is impossible for general problems in case of a continuous input space, which was also discussed in Section 4.2.2. Therefore, the *relaxed problem statement* is introduced, which reads

$$\begin{aligned}
 & \underset{I_1, I_2, \dots, I_d}{\text{maximize}} && \mu(\Omega) \\
 & \text{s.t.} && P\left(\tilde{a}_l < \tilde{a} < \tilde{a}_u \mid \tilde{N}_g, \tilde{N}\right) > 1 - \alpha_c \\
 & && \Omega \subseteq \Omega_{ds}
 \end{aligned} \tag{4.9}$$

with $\mu(\Omega)$ as defined in Equation (4.2). \tilde{a} is the *true fraction of good designs* in the box Ω and \tilde{a}_l and \tilde{a}_u are the *lower and upper bounds* of \tilde{a} . $1 - \alpha_c$ is the *critical confidence level* for the probability denoted by P that \tilde{a} is within the confidence interval $[\tilde{a}_l, \tilde{a}_u]$. \tilde{N}_g is the *number of good designs* out of \tilde{N} sample points. By means of Bayesian statistics,

it is shown that if 100 out of 100 randomly distributed sample points are good designs the probability of \bar{a} lying between 97% and 100% is 95%, independently of the number of dimensions; see [78] and [45].

4.4.2 Algorithm

The algorithm has two main phases, the *Exploration Phase* and the *Consolidation Phase*. In the Exploration Phase the box moves towards a region of good designs, where the box size measure is maximum, while in the Consolidation Phase the fraction of good designs within the box is increased in order to satisfy the constraint of problem statement (4.9). The procedure is as follows:

1. Identify a good design by classical optimization; see Figure 1.9.
2. Construct an initial box around the good design with zero volume.
3. Extend the box into all dimensions.
4. Create uniformly distributed sample points within the box (= Monte Carlo sampling), evaluate each point w.r.t. its performance (= regarding its Solution Space constraints) and categorize each point as good or bad.
5. Apply the *Trim Algorithm*: Within three nested loops over all good designs (outer loop), over all bad designs and over all dimensions (inner loop), the boundaries of the box are moved (details are not mentioned) such that a box with good designs only (= solution box) is obtained. For each outer loop, a candidate solution box is constructed, and hence the number of obtained boxes is equal to the number of good designs. The box with maximum size measure is selected for the next step.
6. If the size measure changes compared to the size measures of the last iterations, repeat steps (3) to (6), otherwise proceed with the next step.
7. Repeat step (4).
8. If the probability in problem statement (4.9) is beyond a specified threshold, repeat step (5), (4) and (8), otherwise stop.

The algorithm is extended in [30] by setting additional constraints in order to include particular designs in the solution box.

4.4.3 Properties of the algorithm

The numerical complexity of creating a Monte Carlo sample and computing the performance for each sample point is $O(\tilde{N})$. For the *Trim Algorithm* it is $O(\tilde{N}^2 d)$.

In [36], the results of 100 runs with the stochastic algorithm with 50 and 100 sample points are compared to the analytically calculated size measure of the optimal box. The algorithm stops in the Consolidation Phase, if 100 out of 100 sample points in a specified number of subsequent iterations are good designs. For multiple test problems (two-dimensional Rosenbrock (nonlinear), two- and three-dimensional convex polytope (linear), d -dimensional Hyperbox and Tilted-Hyperplane with $d = 2, 10, 20, \dots, 100$ (linear)), the standard deviations of the numerical results as well as the absolute and relative errors between the analytic solutions and the means of the numerical solutions are considered. The main conclusions are:

- The standard deviation as well as the error decreases with an increasing number of sample points.
- For the two- and three-dimensional examples, the mean of the numerical solutions approximately agrees with the analytic solutions.
- In the case where the boundaries of the Solution Space constraints are parallel to the edges of the box (Hyperbox problem; see Figure 4.7), the error is low even in high dimensions. Note that contrary to the case where the boundaries are not parallel to the edges of the box the volume of the box is lower compared to the analytic solution; see Figure 4.8.
- In the case where the boundaries of the Solution Space constraints are not parallel to the edges of the box (Tilted-Hyperplane problem), the error strongly increases with an increasing number of dimensions, while the fraction of good designs within the box is still close to 100%. This effect is called *Vertex Problem* and is summarized below.
- The number of necessary iterations for satisfying the constraint of optimization problem (4.9) in the Consolidation Phase increases when the number of dimensions increases. In the case where the boundaries of the Solution Space constraints are parallel to the edges of the box, the algorithm converges faster than in cases where the boundaries of the Solution Space constraints are not parallel to the edges of the box.

Figure 4.7 depicts the convergence of the algorithm within the Consolidation Phase depending on the number of dimensions. As mentioned above, the algorithm terminates if all sample points in a specified number of subsequent iterations are good, here 100 out of 100, i.e. $\tilde{N}_g/\tilde{N} = 1$. On the left side, the boundaries of the Solution Space are parallel to the edges of the box (Hyperbox problem), while they are not parallel in the example on the right side (Tilted-Hyperplane problem).

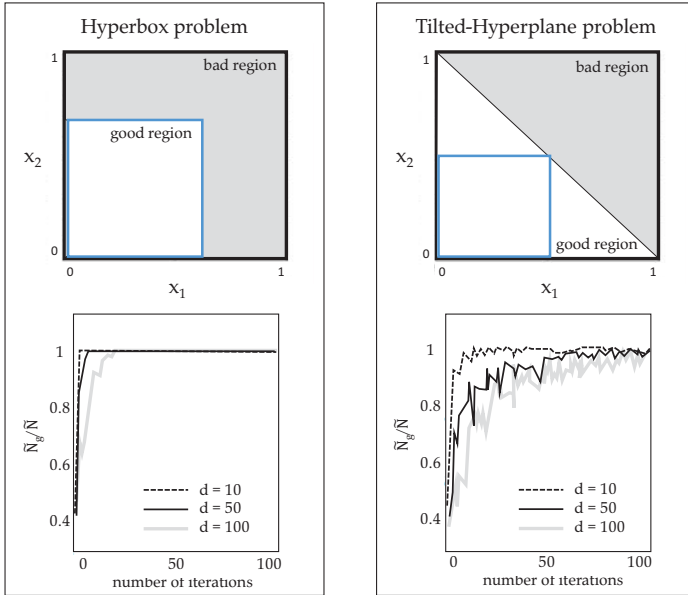


Figure 4.7: The convergence behavior of the stochastic Solution Space algorithm for the Hyperbox and Tilted-Hyperplane problem with 100 sample points. Bottom: The graphs, taken from [36], show the ratio of good designs out of 100 sample points within the box in dependence on the iteration steps for the Hyperbox problem (left) and for the Tilted-Hyperplane problem (right) (Consolidation Phase only).

Vertex Problem The Vertex Problem (in [36] it's called *Corner Problem*) appears for problems where the edges of the box are not parallel to the boundaries of the Solution Space constraints and depends on both the angle between the edges and the boundary as well as on the shape of the boundaries. The effect is shown in [36] by computing the ratio of the average of the size measures of the solution boxes obtained by 100 runs with the stochastic algorithm to the size measure of the analytic result. For instance, it is shown that for the Tilted-Hyperplane problem the ratio increases significantly with increasing number of dimensions; see Figure 4.8. Nevertheless, note that for all boxes obtained in [36] it holds true that the true fraction of good designs lies between 97% and 100% with a 95% confidence level.

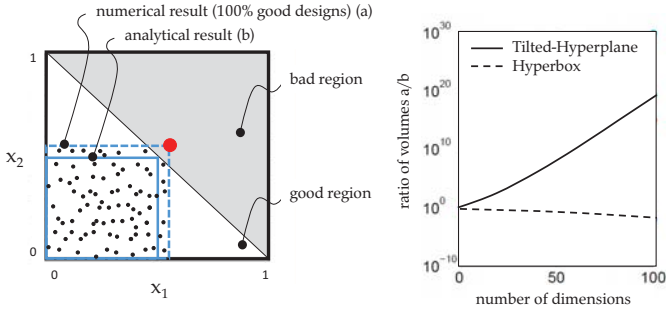


Figure 4.8: Left: The Tilted-Hyperplane problem in two dimensions with the optimal box and an increased box with 100 out of 100 good sample points. Right: The graph, taken from [36], shows the ratio of the average size measure of a box obtained from 100 runs with the stochastic algorithm to the size measure of the analytic result for an increasing number of dimensions.

4.5 Tracking the vertexes of a box: Linearly constrained Solution Spaces

In this section, the general problem statement for seeking box-shaped Solution Spaces with maximum box size measure is reformulated such that it can be solved by any standard optimization algorithm. The approach of vertex tracking is applied and the Solution Space constraints are reformulated w.r.t. the lower and upper boundaries of the i -th interval, which are the optimization parameters. The complete Solution Space is assumed to be specified by a set of linear inequalities (*Type A*; see Figure 3.4).

4.5.1 Problem statement

For a linearly constrained complete Solution Space, the problem statement (4.1) reads

$$\begin{aligned} & \underset{I_1, I_2, \dots, I_d}{\text{maximize}} && \mu(\Omega) \\ & \text{s.t.} && \mathbf{G}\mathbf{x} \leq \mathbf{g}_c \quad \forall \mathbf{x} \in \Omega \subseteq \Omega_{\text{ds}}. \end{aligned} \quad (4.10)$$

The **coefficients of the linear inequalities** are given by $\mathbf{G} \in \mathbb{R}^{m \times d}$ with elements G_{ji} , $i = 1, \dots, d$, $j = 1, \dots, m$. The **thresholds** are stated by $\mathbf{g}_c \in \mathbb{R}^m$. For solving optimization problem (4.10), the optimization constraints must be reformulated as inequalities w.r.t. the optimization parameters, which are the intervals, specified by lower and upper bounds for each design variable x_i^l and x_i^u . The box is defined as the Cartesian product of intervals $\Omega = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \dots \times [x_d^l, x_d^u]$. For better readability, the **box vector** is

introduced and defined as

$$\boldsymbol{\xi} = (\mathbf{x}^l, \mathbf{x}^u)^T \in \mathbb{R}^{2d}. \quad (4.11)$$

In the case of linear inequalities, vertex tracking can be applied; see Section 4.2.2. For linear functions, each constraint can be assigned to a particular vertex of the box; see also [26] and [27]. Hence, only m out of 2^d vertexes must be assessed to ensure that the box contains good designs only. The optimization problem can be rewritten as follows:

$$\begin{aligned} & \underset{\boldsymbol{\xi}}{\text{minimize}} && -\ln \mu(\Omega) \\ & \text{s.t.} && \mathbf{\Gamma} \boldsymbol{\xi} \leq \mathbf{g}_c \\ & && \boldsymbol{\xi} \in \Omega_{\text{bs}} \end{aligned} \quad (4.12)$$

with Ω_{bs} as the **box-space**, which is $\Omega_{\text{ds}} \times \Omega_{\text{ds}}$. The size measure $\mu(\Omega)$ written w.r.t. $\boldsymbol{\xi}$ reads

$$\mu(\Omega) = \prod_{i=1}^d (\xi_{i+d} - \xi_i). \quad (4.13)$$

To ensure positive values for the box size measure only, the additional constraints must be considered:

$$\begin{aligned} & \xi_i < \xi_{i+d} \\ & \forall i = 1, \dots, d \end{aligned} \quad (4.14)$$

which forces that the upper boundary must be larger than the lower boundary for each design variable. Each constraint is assigned to an appropriate vertex. Therefore, $\mathbf{\Gamma} \in \mathbb{R}^{m \times 2d}$ is introduced, which assigns the elements of \mathbf{G} to the box variables $\boldsymbol{\xi}$:

$$\mathbf{\Gamma} = [\mathbf{\Gamma}_1, \mathbf{\Gamma}_2] \in \mathbb{R}^{m \times 2d}. \quad (4.15)$$

One can derive $\mathbf{\Gamma}$ from \mathbf{G} as:

$$\Gamma_{1,ji} = \begin{cases} 0 & \text{for } G_{ji} \geq 0 \\ G_{ji} & \text{otherwise} \end{cases}, \quad \Gamma_{2,ji} = \begin{cases} G_{ji} & \text{for } G_{ji} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.16)$$

with $i = 1, \dots, d$ and $j = 1, \dots, m$.

The reformulation as a minimization problem of the negative logarithm of the size measure instead of the maximization of the size measure as stated in (4.1) yields a convex optimization problem, see proof below. This implies that a local minimum is also the global minimum and, effective computational methods exist to solve these types of optimization problems; see [13]. Additionally, the logarithm acts as a barrier for $\mu(\Omega) \rightarrow 0$ whereby very small interval widths are avoided. The latter is motivated by the fact that generally narrow intervals mean no robustness or no flexibility, which is not desired in practice.

Derivatives of the objective function The first derivatives of the objective function are

$$\frac{\partial(-\ln \mu(\Omega))}{\partial \xi_i} = -\frac{\partial(-\ln \mu(\Omega))}{\partial \xi_{i+d}} = (\xi_{i+d} - \xi_i)^{-1} \quad (4.17)$$

$$i = 1, \dots, d.$$

The second derivatives of the objective function are

$$\begin{aligned} \frac{\partial^2(-\ln \mu(\Omega))}{\partial \xi_i^2} &= \frac{\partial^2(-\ln \mu(\Omega))}{\partial \xi_{i+d}^2} = \\ &= -\frac{\partial^2(-\ln \mu(\Omega))}{\partial \xi_{i+d} \partial \xi_i} = -\frac{\partial^2(-\ln \mu(\Omega))}{\partial \xi_i \partial \xi_{i+d}} = (\xi_{i+d} - \xi_i)^{-2} \end{aligned} \quad (4.18)$$

$$i = 1, \dots, d.$$

All other entries of the Hessian are zero.

Properties of the problem Considering Equation (4.17) and assuming $\xi_i < \xi_{i+d}$, see condition (4.14), show that the objective function of the optimization problem (4.12) is monotonously increasing w.r.t. ξ_i , i.e. the lower boundary of an interval (the first derivative is positive), and monotonously decreasing w.r.t. ξ_{i+d} , i.e. the upper boundary of an interval (the first derivative is negative).

The convexity of the objective function can be shown by the following consideration (in accordance with [60]):

$$-\ln \mu(\Omega) = -\ln \prod_{i=1}^d (\xi_{i+d} - \xi_i) = \sum_{i=1}^d -\ln (\xi_{i+d} - \xi_i). \quad (4.19)$$

The derivatives of each term of the sum are

$$\begin{pmatrix} \frac{\partial(-\ln(\xi_{i+d}-\xi_i))}{\partial \xi_i} \\ \frac{\partial(-\ln(\xi_{i+d}-\xi_i))}{\partial \xi_{i+d}} \end{pmatrix} = \begin{pmatrix} (\xi_{i+d} - \xi_i)^{-1} \\ -(\xi_{i+d} - \xi_i)^{-1} \end{pmatrix} \quad (4.20)$$

with the second derivatives as

$$\mathbf{H} = \begin{pmatrix} (\xi_{i+d} - \xi_i)^{-2} & -(\xi_{i+d} - \xi_i)^{-2} \\ -(\xi_{i+d} - \xi_i)^{-2} & (\xi_{i+d} - \xi_i)^{-2} \end{pmatrix}. \quad (4.21)$$

The eigenvalues λ are obtained by solving the equation $\det(\mathbf{H} - \lambda \mathbf{I}) = 0$ with \mathbf{I} as the

2×2 identity matrix; see Definition 8:

$$((\xi_{i+d} - \xi_i)^{-2} - \lambda)^2 - (-(\xi_{i+d} - \xi_i)^{-2})^2 = \lambda(\lambda - 2(\xi_{i+d} - \xi_i)^{-2}) = 0. \quad (4.22)$$

The solution is

$$\lambda_1 = 0 \text{ and } \lambda_2 = 2(\xi_{i+d} - \xi_i)^{-2}. \quad (4.23)$$

For $\xi_i < \xi_{i+d}$ the eigenvalues are positive, and hence the terms of the sum in Equation (4.19) are convex w.r.t. ξ_i and ξ_{i+d} . As the sum of convex functions is convex, the objective function is shown to be convex. In conjunction with the linear constraints of the optimization problem (4.12), this implies a convex optimization problem.

Remark. To show that $\mu(\Omega)$ is neither concave nor convex the two-dimensional case with two design variables and four interval boundaries is used as a counter example, i.e. $\mu(\Omega) = (\xi_3 - \xi_1)(\xi_4 - \xi_2)$. The Hessian is

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{pmatrix}.$$

The eigenvalues are $\lambda_{1,2} = 0$, $\lambda_3 = 2$ and $\lambda_4 = -2$, i.e. the Hessian is indefinite, and hence $\mu(\Omega)$ is neither concave nor convex.

Remark. Boyd [13] states, that the logarithm of the volume of a polytope described by a set of linear inequalities $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ is a concave function w.r.t. the thresholds of the inequalities \mathbf{b} . The polytope here is the box, and the thresholds are the lower and upper boundaries of the intervals. Thus, the logarithm of the volume of the box is concave and consequently, the negative logarithm of the volume of the box $-\ln \mu(\Omega)$ is convex. This statement is used and described in more detail in Section 4.6.1.

Computation of the outer box In the case of linear Solution Space constraints, the complete Solution Space is a d -dimensional polytope. For the computation of the outer box, the vertexes of the polytope are calculated. Therefore, $\binom{m+2d}{d} = \prod_{i=1}^d \frac{m+2d+1-i}{i}$ (= Binomial coefficient, all possible combinations) number of linear systems are solved to obtain all possible intersection points of the boundaries of the linear Solution Space constraint functions, i.e. $\mathbf{g}(\mathbf{x}) = \mathbf{g}_c$ and the boundaries of the design space. Those intersection points that satisfy the inequalities of the optimization problem (4.10) are sought. As this might get expensive for a large number of Solution Space constraints m or a large number of design variables d , alternatively, the outer box can be obtained by solving optimization problem (4.29) and optimization problem (4.30) instead.

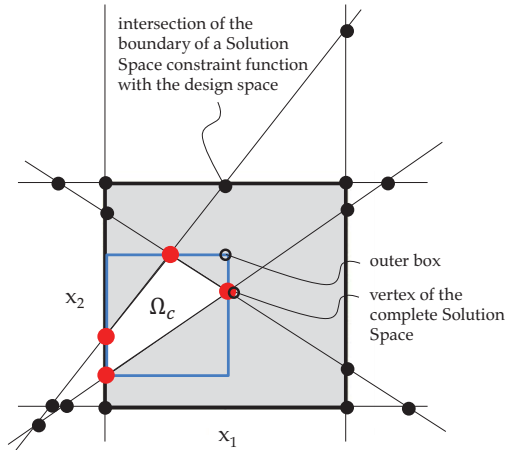


Figure 4.9: Determination of the vertices of the complete Solution Space (specifying the outer box) depicted by a two-dimensional example with $d = 2$ and $m = 3$. Note that the number of possible intersections is 21, however, those boundaries of the design space, which are parallel, have no intersection point.

4.5.2 Implementation

Optimization algorithm Since the gradient information is available and the optimization problem is convex, a gradient-based optimization algorithm for fast convergence is preferred. Involving linear inequality constraints, the problem (4.12) in conjunction with (4.13) and (4.14) is solved by an interior-point algorithm, namely the implementation in MATLAB[®], called by the command `fmincon()`. For further information about the implementation of the algorithm see [15, 16, 71]. Fundamentals of interior-point algorithms are provided in Appendix A.2.

Determination of initial values In order to provide suitable start values for the optimization parameters, an initial box is determined by the following procedure: Firstly, any design that satisfies all Solution Space constraints is selected as the center of the initial box. The design can be found by solving e.g. the following optimization problem: $\min_{\alpha, \mathbf{x}} \alpha$ s.t. $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c + \alpha$. The edge lengths of the box (= interval width) can be selected by e.g. 10% of the design space. If the box contains bad designs, the edge lengths of the initial box are iteratively decreased until the box contains good designs only. Note that the interior-point algorithm implemented in MATLAB[®] does not require initial

values that satisfy all the optimization constraints. However, providing infeasible initial values requires additional iteration steps.

4.5.3 Numerical results

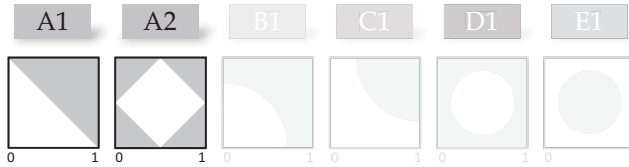


Figure 4.10: Overview of the test problems for $d = 2$. Only the linear examples are considered.

In order to examine the numerical effort in dependence on the number of dimensions d and the number of Solution Space constraints m , the optimization problem (4.12) is solved for the proposed linear test problems listed in Table 3.1. The design space for the examples considered here is given by $\Omega_{\text{ds}} = [0, 1]^d$. For a better comparison of the results, the initial box is given and is $\Omega = [0.01, 0.11]^d$ for test problem *A1* and $\Omega = [0.45, 0.55]^d$ for test problem *A2*. Table 4.1 shows the number of iterations, the number of function evaluations, i.e. overall evaluations of the objective function and constraints, the CPU time as well as the relative error of the numerical results compared to the analytic solution. For the relative error, the size measure obtained by numerical optimization $\mu(\Omega)_{\text{nn}}$ and the optimal size measure obtained by analytic calculus $\mu(\Omega)_{\text{an}}$ is considered. The relative error is defined as $\frac{\mu(\Omega)_{\text{nn}} - \mu(\Omega)_{\text{an}}}{\mu(\Omega)_{\text{an}}}$. Table 3.2 shows the formula for the optimal size measures of test problem *A1* and *A2*. Technical details about the computer and software used are provided in the Appendix.

Table 4.1: Numerical effort for the computation of the optimal inner box for the linear test problems *A1* and *A2*. While the number of Solution Space constraints is kept constant (for test problem *A1* $m = 1$ and for *A2* $m = 4$), the number of dimensions is increased. The values are associated with the following number of dimensions $d = 2, 4, 10, 50$.

	no. iterations	no. fun. eval.	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
<i>A1</i>	10, 10, 10, 10	13, 12, 12, 12	0.06, 0.10, 0.11, 0.12	-6e-6, -1e-5, -2e-5, -1e-4
<i>A2</i>	8, 9, 9, 10	9, 11, 10, 13	0.07, 0.09, 0.10, 0.12	-8e-6, -8e-6, -8e-6, -8e-6

Additionally, the outer box is computed by the calculation of the vertexes of the polytope, enclosing all good designs as proposed in Section 4.5.1. Table 4.2 shows the number of linear equations to be solved for the computation of the outer box, the CPU time as well as the relative error of the numerical results compared to the analytic solution. Due to a very high number of linear equations of approximately $1e30$ in the case of $d = 50$, the problem of seeking the minimum outer box cannot be solved practicably by applying the approach proposed in Section 4.5.1. Problem (4.29) and (4.30) are solved instead. The same optimizer as for solving the problem of seeking the maximum inner box is used, i.e. the interior-point algorithm implemented in MATLAB®. The numerical results obtained by solving (4.29) and (4.30) are highlighted by an asterisk (*).

Table 4.2: Numerical effort for the computation of the optimal outer box for the linear test problems *A1* and *A2*. While the number of Solution Space constraints is kept constant (for test problem *A1* $m = 1$ and for *A2* $m = 4$) the number of dimensions is increased. The values are associated with the following number of dimensions $d = 2, 4, 10, 50$.

	no. lin. equations	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
<i>A1</i>	10, 126, 3.53e05, 2.00e29	0.002, 0.02, 25.30, 20.64*	0, 0, 0, -2e-2*
<i>A2</i>	28, 495, 1.96e06, 1.47e30	0.004, 0.05, 65.36, 84.64*	0, 0, 0, -1e-3*

* problems (4.29) and (4.30) are solved

For all optimization problems solved by *fmincon()*, the final first-order optimality measure and the final maximum constraint violation are less than the critical threshold of $1e-6$. For more information about the first-order optimality measure see Appendix A.1.

Discussion Table 4.1 shows, that the *number of iterations* is not or only slightly influenced by the number of dimensions for test problem *A1* and *A2*. Furthermore, the *number of function evaluations*, i.e. evaluation of the objective function and constraints, equals approximately the number of iterations. A higher number of function evaluations compared to the number of iterations occurs if the solver attempts a step, and rejects the attempt; see Appendix A.2. The CPU time is roughly a tenth of a second for all number of dimensions and both test problems. The relative error of the numerical result compared to the analytic solution is less than $1e-3\%$ for all test problems and increases with the number of dimensions in case of test problem *A1*.

In summary, both linear test problems can be solved very efficiently and precisely, nearly independently of the number of dimensions.

For the computation of the outer box, the vertexes of the high-dimensional polytope, which encloses the good designs, are calculated. The number of linear equations to be

solved scales with the number of dimensions and the number of constraints; see Section 4.5.1. Consequently, the CPU time increases with an increasing number of linear equations to be solved. For the 50-dimensional examples, the number of linear equations in both examples is approximately $1e30$, which is not practicable to be solved and the approach proposed in Section 4.6 (paragraph *computation of the outer box*) is used instead. The numerical results marked by an asterisk (*) are obtained by applying `fmincon()`, i.e. are solved iteratively. The result shows a relative error compared to the analytic solution. However, the error is very low.

4.6 Tracking the vertexes of a box: Nonlinearly constrained Solution Spaces

In Section 4.5 the approach of vertex tracking was used for a direct evaluation of the box regarding its Solution Space constraints. Therefore, each Solution Space constraint is assigned to a particular vertex of the box and only m function evaluations are necessary to assess the box. In this section, the approach is extended to a broader range of types of Solution Space constraints, particularly monotone functions (*Type B* and *Type C*; see Figure 3.4). A strategy to handle even non-monotone functions (*Type D* and *Type E*) is proposed at the end of this section.

4.6.1 Problem statement

Monotone Solution Space constraints The problem statement for seeking a box with maximum size measure applying vertex tracking reads

$$\begin{aligned} & \underset{\boldsymbol{\xi}}{\text{minimize}} && -\ln \mu(\Omega) \\ & \text{s.t.} && \mathbf{g}(\boldsymbol{\gamma}(\boldsymbol{\xi})) \leq \mathbf{g}_c \\ & && \boldsymbol{\xi} \in \Omega_{\text{bs}} \end{aligned} \quad (4.24)$$

with the box size measure $\mu(\Omega)$ as defined in (4.13), $\boldsymbol{\xi}$ as the box vector; see (4.11) and \mathbf{g} and \mathbf{g}_c as stated in (4.1). Additionally, the inequality constraints (4.14) must be considered. γ_j , $j = 1, \dots, m$ are linear functions, each assigns a Solution Space constraint to a particular vertex of the box. $\gamma_j : \boldsymbol{\xi} \mapsto \boldsymbol{x}$ is defined as

$$\gamma_j(\boldsymbol{\xi}) = \mathbf{C}^j \boldsymbol{\xi} \quad (4.25)$$

with \mathbf{C}^j as the **vertex assignment matrix** of the j -th constraint:

$$\mathbf{C}^j = [\mathbf{C}_1^j, \mathbf{C}_2^j] \in \mathbb{R}^{d \times 2d}. \quad (4.26)$$

One can derive \mathbf{C}^j from $\tilde{\mathbf{G}}$ as:

$$C_{1,ii}^j = \begin{cases} 0 & \text{for } \tilde{G}_{ji} \geq 0 \\ 1 & \text{otherwise} \end{cases}, \quad C_{2,ii}^j = \begin{cases} 1 & \text{for } \tilde{G}_{ji} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.27)$$

with $i = 1, \dots, d$. All other entries of \mathbf{C}^j are equal to zero. Only the algebraic sign of the entries of $\tilde{\mathbf{G}}$ denoted by \tilde{G}_{ji} are of interest. $\tilde{\mathbf{G}}$ can be obtained by linearizing \mathbf{g} at any point \mathbf{x} . Hence, only $d + 1$ additional function evaluations for each of the m constraints are necessary.

$$\tilde{G}_{ji} \approx \frac{g_j(\mathbf{x}_{\sim i}) - g_j(\mathbf{x})}{\delta x_i} \quad (4.28)$$

$\mathbf{x}, \mathbf{x}_{\sim i} \in \Omega_{\text{ds}}$

with $\mathbf{x}_{\sim i} = \mathbf{x} + \delta x_i \mathbf{e}_i$ as an arbitrary design, where the i -th component is shifted by δx_i .

Non-monotone Solution Space constraints If a Solution Space constraint function g_j is *non-monotone* w.r.t. some or all design variables, i.e. does not satisfy (3.6) and (3.7), but is *convex* (*Type D*, see Figure 3.4), i.e. satisfies (3.8), vertex tracking can be applied. However, in contrast to the case of monotone Solution Space constraints, each constraint must be assigned to *multiple vertexes*. $\Psi = (\Psi_1, \Psi_2, \dots, \Psi_d)$ is the set of indices of those design variables for which g_j shows non-monotone but convex behavior. The number of elements of Ψ is \tilde{d} . The vertexes per constraint, which must be evaluated, are $\{x_{\Psi_1}^l, x_{\Psi_1}^u\} \times \{x_{\Psi_2}^l, x_{\Psi_2}^u\} \times \dots \times \{x_{\Psi_{\tilde{d}}}^l, x_{\Psi_{\tilde{d}}}^u\}$ and hence the number of vertexes is $2^{\tilde{d}}$. If g_j is a monotone function w.r.t. all design variables, \tilde{d} is zero and the number of vertexes is one, as stated above (paragraph *monotone Solution Space constraints*). For $\tilde{d} = d$ all vertexes of the box must be evaluated, compare Figure 4.2 case (C). This is repeated for each Solution Space constraint, whereas a maximum of $m2^{\tilde{d}}$ function evaluations are necessary to check the m constraints of optimization problem (4.1). For Solution Space constraints of *Type E*, condition (4.8) must be checked. If the condition is not satisfied, vertex tracking cannot be applied and in the presence of a continuous input space, a direct evaluation of the box, see Section 4.2.2, is impossible. Consequently, an indirect evaluation is necessary, e.g. by sampling and statistical assessment.

Properties of the problem As shown in Section 4.5.1 the objective function $-\ln \mu(\Omega)$ is monotone and convex. However, in case of non-convex constraints the optimization problem is not convex and local minima may occur.

Computation of the outer box For the computation of the outer box, the following problem can be solved by any standard optimizer:

$$\begin{aligned} & \underset{\mathbf{x}}{\text{maximize}} && x_i \\ \text{s.t.} &&& \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c \\ &&& \mathbf{x} \in \Omega_{\text{ds}}. \end{aligned} \tag{4.29}$$

Solving (4.29) yields the maximum value of x_i , while all the other design variables are adjusted appropriately. To obtain the minimum value of x_i , the following optimization problem must be solved:

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && x_i \\ \text{s.t.} &&& \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c \\ &&& \mathbf{x} \in \Omega_{\text{ds}}. \end{aligned} \tag{4.30}$$

In the presence of particular types of Solution Space constraint functions, solving the problems (4.29) and (4.30) for all design variables yields the outer box as defined in (4.4). The objective function is linear, however, the constraints might be nonlinear with concave or neither concave nor convex behavior. The latter yields an optimization problem with local minima; non-connected domains are possible; see Figure 4.11. Consequently, the result of the optimization depends on the initial values and a globalization strategy is recommended, e.g. running the optimization multiple times with randomly selected initial guesses or using global optimization algorithms, e.g. evolutionary algorithms. Note that in case of bad regions enclosed by the Solution Space, case (D) in Figure 4.11, solving (4.29) and (4.30) will provide a box that is equal to the design space.

4.6.2 Implementation

Optimization algorithm Depending on whether the Solution Space constraints are given as parametric or oracle functions, the derivatives can be provided to the optimizer or must be estimated by e.g. finite-differences. However, since at least the derivatives of the objective function are available, see Equation (4.17) and (4.18), a gradient-based optimizer for fast convergence is preferred, namely the interior-point implementation in the MATLAB® environment; see also Section 4.5.2.

For seeking the minimum outer box, problems (4.29) and (4.30) are solved by applying the same optimizer as for solving the problem of seeking the maximum inner box (4.24), i.e. the interior-point algorithm.

Vertex tracking For the approach of vertex tracking in the case of oracle functions, where nothing is known about the structure of the equations, the types of the Solution Space constraint functions must be determined by checking conditions (3.6), (3.7) and

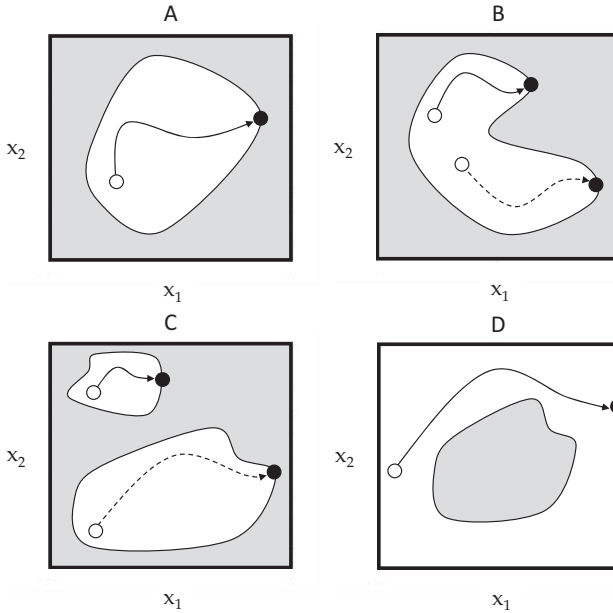


Figure 4.11: Determination of the maximum value of x_1 by solving optimization problem (4.29). In the case of a convex domain (A), the optimization result is globally optimal and does not depend on the initial value. If the domain is neither convex nor concave (B) or not connected (C) the result may depend on the initial value (in dependence on the optimization algorithm (local, global strategies)). In the case of bad regions enclosed by the Solution Space (D), solving problem (4.29) will yield a box that is equal to the design space.

(3.8), respectively, in conjunction with sampling. Therefore, the statement *for all \mathbf{x}* can only be assessed statistically, an example is provided in Section 7.1.4.

Determination of initial values For the determination of initial values see Section 4.5.2.

4.6.3 Numerical results

In order to examine the numerical effort in dependence on the number of dimensions d , optimization problem (4.24) is solved for the proposed monotone and non-monotone but convex test problems. The design space for test problem *B1* (monotone, convex), *C1*

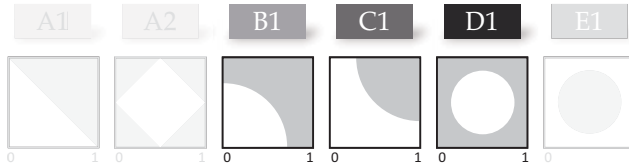


Figure 4.12: Overview of the test problems for $d = 2$. Only the nonlinear monotone and non-monotone but convex test problems are considered, due to the requirements of the approach.

(monotone, non-convex) and $D1$ (non-monotone, convex) is given by $\Omega_{ds} = [0, 1]^d$. For test problem $B1$ and $C1$, the initial box is $\Omega = [0.01, 0.11]^d$ and for $D1$ it is $\Omega = [0.45, 0.55]^d$. Table 4.3 shows the number of iterations, the number of function evaluations, the CPU time as well as the relative error of the numerical results compared to the analytic solution for the monotone and non-monotone but convex test problems listed in Table 3.1. The relative error is defined as $\frac{\mu(\Omega)_{\text{nu}} - \mu(\Omega)_{\text{an}}}{\mu(\Omega)_{\text{an}}}$, with the size measure obtained by numerical optimization $\mu(\Omega)_{\text{nu}}$ and the optimal size measure obtained by analytic calculus $\mu(\Omega)_{\text{an}}$; see Table 3.2. Technical details about the computer and software used are provided in the Appendix.

Table 4.3: Numerical effort for the computation of the optimal inner box for the monotone ($B1$, $C1$) and non-monotone but convex ($D1$) test problems. While the problems comprise one single Solution Space constraint the number of dimensions is increased. The values are associated with the following number of dimensions $d = 2, 4, 10, 50$.

	no. iterations	no. fun. eval.	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
$B1$	8, 9, 9, 9	10, 12, 11, 10	0.08, 0.09, 0.10, 0.12	-1e-4, -1e-5, -2e-5, -1e-4
$C1$	10, 10, 9, 9	11, 12, 11, 10	0.14, 0.16, 0.15, 0.16	-6e-4, -1e-3, -2e-3, -1e-4
$D1$	7, 7, 24, -	8, 8, 33, -	0.07, 0.25, 15.92, -	-8e-6, -7e-4, -2e-1, -

Additionally, the outer box is computed by solving optimization problems (4.29) and (4.30). Table 4.4 shows the number of optimization problems to be solved for the computation of the minimum outer box, the CPU time as well as the relative error of the numerical results compared to the analytic solution.

All optimization runs terminated due to a final first-order optimality measure, see Appendix A.1, less than the critical threshold of $1e-6$, and the maximum constraint violation is less than the critical threshold of $1e-6$.

Table 4.4: Numerical effort for the computation of the optimal outer box for the monotone ($B1$, $C1$) and non-monotone but convex ($D1$) test problems. While the problems comprise one single Solution Space constraint, the number of dimensions is increased. The values are associated with the following number of dimensions $d = 2, 4, 10, 50$.

	no. opt. problems	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
$B1$	4, 8, 20, 100	0.36, 1.17, 3.20, 21.51	-1e-3, -1e-3, -4e-3, -2e-2
$C1$	4, 8, 20, 100	0.33, 0.91, 3.86, 33.45	-4e-4, -2e-3, -4e-3, -2e-2
$D1$	4, 8, 20, 100	0.32, 0.74, 3.61, 16.98	-1e-3, -2e-4, -2e-4, -1e-3

Discussion Table 4.3 shows that, on average, the *number of iterations* does not increase with an increasing number of dimensions for the monotone test problems $B1$ and $C1$. In the case of the non-monotone but convex test problem $D1$, the number of optimization constraints scales with 2^d based on the fact that the Solution Space constraint is assigned to all vertexes of the box. For test problem $D1$, the number of iterations increases significantly with the number of dimensions and hence the number of optimization constraints. In case of $d = 50$, the number of optimization constraints to be considered is too large to be solved practicably. Furthermore, the *number of function evaluations*, i.e. evaluation of the objective function and constraints, equals approximately the number of iterations, except for the 10-dimensional non-monotone convex test problem. A higher number of function evaluations compared to the number of iterations occurs if the solver attempts a step and rejects the attempt. Independent of the number of dimensions, the CPU time is very low for all test problems, except for the 10-dimensional non-monotone but convex test problem, due to the high number of optimization constraints. The relative error for all test problems is less than 1%.

In summary, in the case of monotone Solution Space constraints, the problem of seeking a box with maximum box size measure can be solved very efficiently, also for a large number of dimensions. The results are very accurate, close to the analytic solution. In the case of a moderate number of dimensions, the non-monotone but convex test problem can be solved with a low computational effort, however, for a high number of dimensions the numerical effort increases significantly due to a very large number of optimization constraints.

For the computation of the outer box, the optimization problems (4.29) and (4.30) are solved for each dimension separately. The CPU time for solving (4.29) and (4.30), respectively is very low and the overall computation time scales linearly with the number of dimensions. The relative error for all test problems is less than 1e-1%.

4.7 Modifications of the optimization problem

As discussed in Section 4.2.1, the inner box provides a weak statement about the infeasibility: There might be more good designs outside that box. In general the number of boxes that satisfy the criteria of an inner box as defined in (4.3) is infinite. In industry often apriori information is available, e.g. particular designs are preferred and should be included in the box if possible, some design variables show a higher degree of uncertainty and a larger width of the associated interval is desired, etc. For that reason, the problem statement for seeking a box with maximum size measure can be extended by the following constraints:

1. The lower or/and upper boundaries of the box are set to a specified value and may not be changed during optimization (e.g. setting the lower boundary of design variable x_1 to 0.5: $\xi_1 = 0.5$) (*M1*).
2. The lower and/or upper boundaries of the box are constrained by lower and/or upper bounds, i.e. modifications of Ω_{bs} (e.g. bounding the lower boundary of design variable x_1 between values of 0 and 0.8: $0 \leq \xi_1 \leq 0.8$) (*M2*).
3. The width of two or more intervals are constrained to be equal (e.g. the interval width of design variable x_1 and x_2 are constrained to be equal) (*M3*).
4. The widths of intervals are weighted by weighting factors (*M4*).
5. The widths of intervals are constrained to be equal to a specified value or greater/less than a specified value (*M5*).
6. The widths of intervals are constrained to have a value of zero, i.e. values of particular design variables are adjusted such that the product of the intervals of the other design variables is maximum (*M6*).

4.7.1 Implementation

For weighting intervals (see *M4* in Section 4.7) the size measure (4.13) is modified to

$$\mu(\Omega) = \prod_{i=1}^d (\xi_{i+d} - \xi_i)^{\omega_i} \quad (4.31)$$

with $\omega_i \geq 1$ as the weighting factor. For the specification of particular interval widths (see *M5* in Section 4.7), the inequality (4.14) is rewritten as equality

$$\begin{aligned} \xi_i &= \xi_{i+d} - \delta w_i \\ \forall i &= 1, \dots, d \end{aligned} \quad (4.32)$$

with $\delta w_i \geq 0$ as the width of the i -th interval. In cases where the widths of intervals are constrained to be greater or less than a specified value, constraint (4.32) is formulated as an inequality. In order to set the width of an interval to zero (see *M6* in Section 4.7), the size measure (4.13) must be modified to

$$\mu(\Omega) = \prod_{i=1}^d \mu(I_i) \quad (4.33)$$

with

$$\mu(I_i) = \begin{cases} \xi_{i+d} - \xi_i & \text{for } \delta w_i > 0 \\ 1 & \text{otherwise} \end{cases} \quad (4.34)$$

Consequently, the derivative terms of the objective function, see (4.17) and (4.18), w.r.t. ξ_i , which are associated with intervals with $\delta w_i = 0$, are set to zero.

4.7.2 Numerical results

The effect of the modifications provided in Section 4.7 on the result of the optimization problem is demonstrated on the two-dimensional Tilted-Hyperplane problem *A1*. Figure 4.13 shows the results for modifications *M1-M6*. For demonstrating the effect of modification *M1*, the upper boundary of design variable x_1 is specified to be 0.4 throughout the optimization. In modification *M2*, the upper boundary of x_1 is specified to lie between 0.6 and 0.7; for modification *M3* the interval of x_1 is specified to have the same width as the interval of x_2 . The effect of modification *M4* is demonstrated by an example, where the interval width of x_1 is weighted by $\omega_1 = 2$, $\omega_1 = 5$ and $\omega_1 = 10$, respectively. For modification *M5* the interval width of x_1 is specified to be equal to 0.3, and for modification *M6*, the largest interval width for x_1 is sought, which is feasible while adjusting x_2 appropriately.

Discussion Note that all solution boxes in Figure 4.13, with the exception of case *M3*, have a smaller size measure as the optimal box obtained by solving problem (4.12) without additional constraints, however, preferences of decision makers can be taken into account. The necessity for specifying additional constraints arises based on the fact that box-shaped Solution Spaces are able to provide a subset of the complete Solution Space only, and hence more good designs outside the box exist. Setting additional constraints enables decision makers to find a box that contains a particular preferred design, provides maximum robustness and flexibility for particular design variables, etc. For instance, the box obtained by solving (4.12) with modification *M4* provides more robustness for design variable x_1 compared to the box obtained by solving (4.12) without modifications (equivalent to the box in case of *M3*), however, the robustness in design variable x_2 decreases.

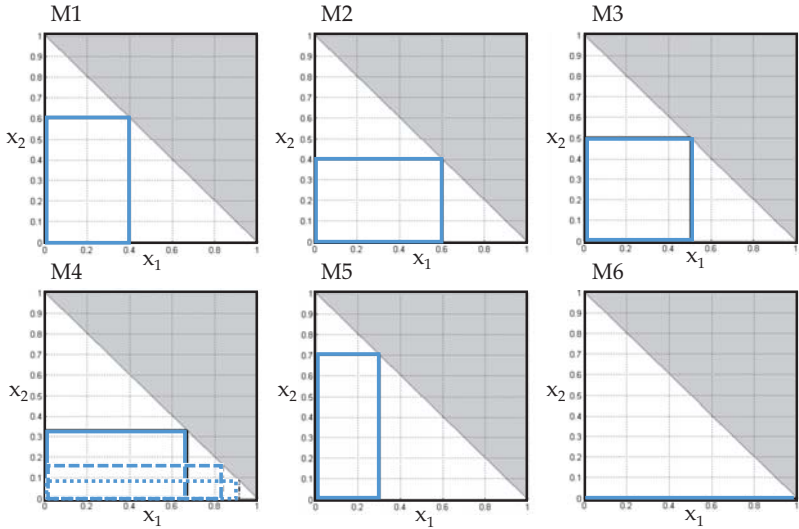


Figure 4.13: Seeking the box with maximum box size measure for the two-dimensional Tilted-Hyperplane problem with modifications *M1-M6*. *M1*: upper boundary of x_1 is specified to be 0.4; *M2*: upper boundary of x_1 is specified to lie between 0.6 and 0.7; *M3*: interval width of x_1 is specified to be the same as the interval width of x_2 ; *M4*: the interval width of x_1 is weighted by a factor of 2, 5 and 10; *M5*: interval width of x_1 is specified to be 0.3; *M6*: seek the largest interval width of x_1 whereas the value of x_2 is adjusted accordingly.

5 — Two-dimensional decomposition of Solution Spaces

In the previous chapter, it was shown that box-shaped Solution Spaces are able to represent high-dimensional Solution Spaces in an intuitive manner and requirements on design variables are fully decoupled. However, this comes with a loss of Solution Space, i.e. the box does not enclose all good designs. In this chapter, another approach is introduced, which decomposes high-dimensional Solution Spaces into 2d-spaces. In order to enclose as many as good designs, the volume of this Solution Space must be maximized. Mathematical formulations of the underlying optimization problems are stated, two approaches are proposed and numerical results based on the analytic test problems are presented.

5.1 2d-spaces as representation of high-dimensional Solution Spaces

Box-shaped Solution Spaces map the requirements on the system onto intervals (completely decoupled $1d$ representation) for each design variable. The Cartesian product of all intervals is a high-dimensional box in the input space. In general, the complete Solution Space has an arbitrary shape and a representation by a box comes with a loss of Solution Space, i.e. the box does not enclose all good designs. However, in many applications it is often not required to decouple requirements on design variables completely. This is the case, if two or more design variables are assigned to a particular component that is designed in detail by one particular group. In the following, design variables are coupled pairwise (2d-spaces), however, the approach proposed can be extended for applications where more than two design variables are coupled. The benefit of coupled requirements on design variables is a reduced loss of Solution Space compared to box-shaped Solution Spaces. Figure 5.1 shows the complete Solution Space of the three-dimensional Tilted-Hyperplane problem, the optimal box as well as the optimal 2d-spaces (with maximum size measure). In the case of the 2d-spaces, the complete Solution Space is expressed as the Cartesian product of a triangle and an interval. However, both approaches are not able to represent the entire set of good designs. Nevertheless, the size of the set represented by 2d-spaces is twice the size represented by a box, i.e. intervals. The statement of box-shaped Solution Spaces, which is

'You can select any value of the design variables within the associated intervals independently, in order to satisfy the requirements on the system',

changes in the case of 2d-spaces to

'You can select any combination of values of two design variables within the associated 2d-space independently in order to satisfy the requirements on the system'.

5.2 General problem statement

Seeking a Cartesian product of two-dimensional Solution Spaces with maximum size measure $\mu(\Omega)$ that contain good designs only reads

$$\begin{aligned} & \underset{\Omega^1, \Omega^2, \dots, \Omega^n}{\text{maximize}} && \mu(\Omega) \\ \text{s.t.} &&& \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c \quad \forall \mathbf{x} \in \Omega \subseteq \Omega_{\text{ds}} \end{aligned} \quad (5.1)$$

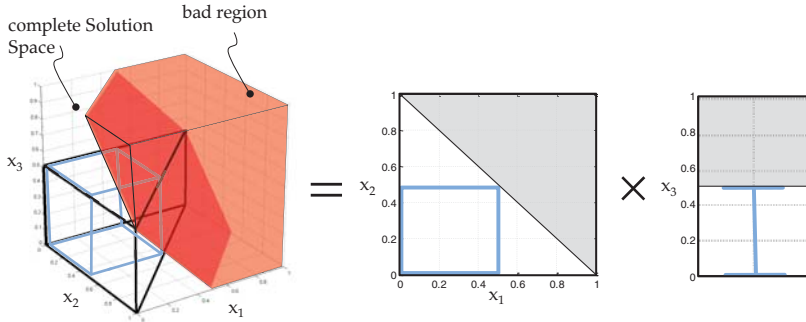


Figure 5.1: A solution box (blue lines) vs. the Solution Space expressed as product of a 2d-space and an interval (white areas) for the Hyperplane-Problem in three dimensions.

with Ω as the Cartesian product of 2d-spaces Ω^k , i.e. $\Omega = \Omega^1 \times \Omega^2 \times \dots \times \Omega^n$ with

$$n = \begin{cases} \frac{d}{2} & \text{for } d \text{ even} \\ \frac{d+1}{2} & \text{otherwise} \end{cases}. \quad (5.2)$$

Without loss of generality concerning $2d$ representations, the design variables are assumed to be coupled pairwise from the first to the last, i.e. the first with the second, the third with the fourth, etc. In case d is odd the n -th 2d-space Ω^n degenerates into an interval. $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$ are the Solution Space constraints as introduced in Section 3.2. Figure 5.2 depicts the set of permissible combinations of pairwise coupled design variables, i.e. 2d-spaces, as white areas. 2d-spaces are subsets of the 2d-design spaces $\Omega_{\text{ds}}^k = [x_{2k-1}^{\text{lb}}, x_{2k-1}^{\text{ub}}] \times [x_{2k}^{\text{lb}}, x_{2k}^{\text{ub}}]$. In accordance with the definition of the box size measure in Section 3.2.4, the *size measure* for 2d-spaces is

$$\mu(\Omega) = \prod_{k=1}^n \mu(\Omega^k) \quad (5.3)$$

with $\mu(\Omega^k)$ as the **measure of the area** of the k -th 2d-space, i.e.

$$\mu(\Omega^k) = \int_{\Omega^k} d\Omega^k \quad (5.4)$$

$$k = 1, \dots, n.$$

In the case where d is odd and hence Ω^n is an interval, $\mu(\Omega^n)$ is the measure of the interval width of the d -th design variable

$$\mu(\Omega^n) = \int_{\Omega^n} d\Omega^n = \int_{x_d^1}^{x_d^u} dx_d = x_d^u - x_d^1. \quad (5.5)$$

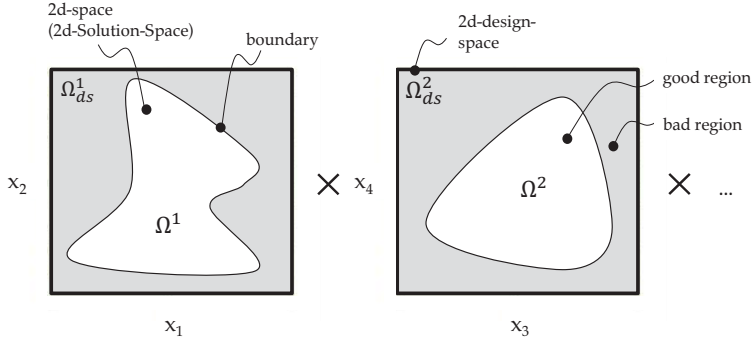


Figure 5.2: 2d-spaces with arbitrary shape. The set of good designs is specified by the Cartesian product of two-dimensional sets $\Omega^1, \Omega^2, \dots, \Omega^n$. Each 2d-space is a subset of the two-dimensional design space Ω_{ds}^k .

In accordance with the modifications of the optimization problem for seeking a box with maximum size measure, see Section 4.7, the objective function of the optimization problems for seeking optimal 2d-spaces can be modified by weighting factors. The size measure for 2d-spaces is modified to

$$\mu(\Omega) = \prod_{k=1}^n \mu(\Omega^k)^{\omega_k} \quad (5.6)$$

with $\omega_k \geq 1$ as the weighting factor.

5.3 Underlying idea

In the following, two approaches that decompose a high-dimensional Solution Space into two-dimensional Solution Spaces are introduced. According to the idea of tracking the vertexes of a box, the first approach is based on the idea of *tracking the vertexes of a high-dimensional polytope*. However, it requires the complete Solution Space to be a convex set and a moderate number of dimensions, since the number of vertexes to be assessed grows

fast with the number of dimensions. The second approach is based on the *decomposition of a set of d -dimensional inequalities into a set of two-dimensional inequalities*, which requires the Solution Space constraints to be decomposable; see Equations (5.7) and (5.8).

Tracking the vertexes of a polytope The set of good designs is specified by the Cartesian product of two-dimensional polygons, for d odd the last dimension is expressed by an interval; see Figure 5.3. If the complete Solution Space is convex, a direct assessment of the enclosed set w.r.t. the Solution Space constraints is possible; see Section 3.4. Hence, it is sufficient to check the vertexes of the polytope in order to guarantee that the enclosed set of designs is strictly good; see Section 5.4. However, the number of vertexes to be checked grows fast with an increasing number of dimensions. If the polygons in all 2d-spaces are assumed to have the same **number of vertexes**, denoted by p , the number of vertexes of the d -dimensional polytope is p^n for d even and $2p^{n-1}$ for d odd. The optimization problem of seeking the optimal position of the vertexes within each 2d-design space yielding a maximum value of $\mu(\Omega)$ is presented in Section 5.4.2.

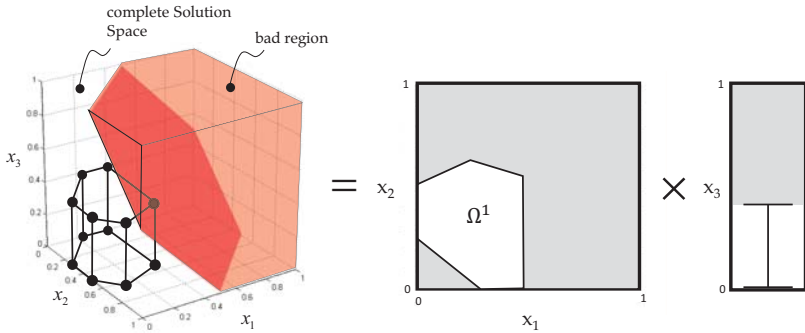


Figure 5.3: A three-dimensional Solution Space (Tilted-Hyperplane problem) expressed as the Cartesian product of a two-dimensional polygon with six vertexes and a one-dimensional interval. To ensure that the enclosed set of designs is good, the vertexes of the polytope are assessed with regard to the Solution Space constraint.

Decomposing Solution Space constraints In order to decouple the requirements pairwise, each of the m Solution Space constraint functions g_j is decomposed as follows:

$$g_j(\mathbf{x}) = g_j^1(x_1, x_2) + g_j^2(x_3, x_4) + \dots + g_j^n(x_{2n-1}, x_{2n}) \quad (5.7)$$

for d even and

$$g_j(\mathbf{x}) = g_j^1(x_1, x_2) + g_j^2(x_3, x_4) + \dots + g_j^{n-1}(x_{2n-3}, x_{2n-2}) + g_j^n(x_d) \quad (5.8)$$

for d odd, $j = 1, \dots, m$. Each two-dimensional set is specified by inequalities in the form $g_j^k(x_{2k-1}, x_{2k}) \leq g_{c,j}^k$ (d even: $k = 1, \dots, n$; d odd: $k = 1, \dots, n-1$ and additionally $g_j^n(x_d) \leq g_{c,j}^n$) with $g_{c,j}^k$ as the **threshold of** the j -th constraint of the k -th 2d-space; see Figure 5.4. In order to guarantee that a design, with all pairs of design variables within their associated 2d-spaces, satisfies the overall requirements on the system, the additional equality must be considered:

$$\sum_{k=1}^n g_{c,j}^k = g_{c,j} \quad (5.9)$$

$$\forall j = 1, \dots, m.$$

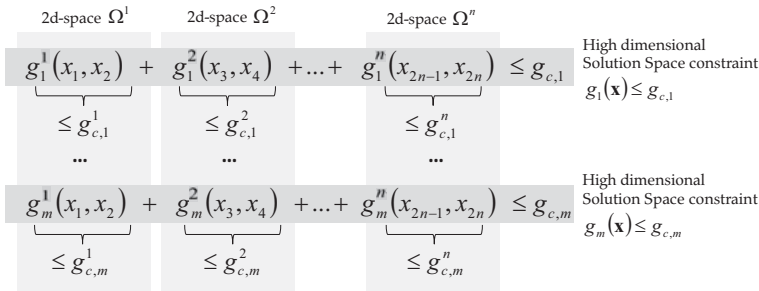


Figure 5.4: Two-dimensional decomposition of m d -dimensional Solution Space constraints. The requirement on each of the Solution Space constraints is decomposed into requirements on two-dimensional functions. The set of two-dimensional inequalities for a pair of design variables is the mathematical description of a 2d-space.

Although an infinite number of values for $g_{c,j}^k$ that satisfy Equation (5.9) exist, some sets of values yield 2d-spaces, representing more good designs than others. This is shown by the four-dimensional Tilted-Hyperplane problem with two sets of $g_{c,j}^k$, both satisfying Equation (5.9): (A) $g_{c,1}^1 = 1, g_{c,1}^2 = 1$ and (B) $g_{c,1}^1 = 1.9, g_{c,1}^2 = 0.1$. Figure 5.5 shows the associated 2d-spaces. Considering the size measure as defined in (5.3) reveals that case (A) represents more good designs ($0.5 \cdot 0.5 = 0.25$) than case (B) ($0.995 \cdot 0.005 = 4.975e-3$). The optimization problem of seeking the set of $g_{c,j}^k$ that maximizes $\mu(\Omega)$ under consideration of (5.9) is presented in Section 5.5.1.

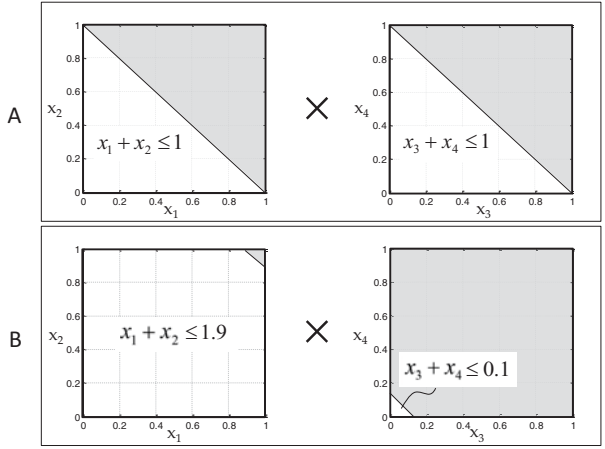


Figure 5.5: Solution Space expressed as a product of two 2d-spaces (white areas) for the Tilted-Hyperplane problem in four dimensions for two different sets of values for the thresholds of the two-dimensional inequalities.

5.4 Tracking the vertexes of a polytope

In this section, the idea of tracking the vertexes of a d -dimensional polytope is used to reformulate the general problem statement (5.1) such that it can be solved by any standard optimization algorithm. In the following, the complete Solution Space as defined by (3.1) is assumed to be a convex set. Without loss of generality in the following, it is assumed that each two-dimensional polygon has the same number of vertexes. Furthermore, the design variables are coupled pairwise from the first to the last, i.e. the first with the second, the third with the fourth, etc.

5.4.1 Assessing polytope-shaped sets regarding Solution Space constraints

As a condition for vertex tracking in conjunction with polytopes, the definition of sublevel sets for box-shaped Solution Spaces, see Definition 11, is modified; see also Figure 5.6.

Definition 12. *In case of polytopes the two-dimensional sublevel set $\mathcal{S}_{g_c,k}(\mathbf{g}(\mathbf{x}))$ is the*

set of good designs enclosed by a polygon in the k -th 2d-space:

$$\mathcal{S}_{g_c, k}(\mathbf{g}(\mathbf{x})) = \{(x_{2k-1}, x_{2k}) \in [x_{2k-1}^{lb}, x_{2k-1}^{ub}] \times [x_{2k}^{lb}, x_{2k}^{ub}] \mid \mathbf{g}(\mathbf{x}_{\sim k}) \leq \mathbf{g}_c\} \quad (5.10)$$

$$k = 1, \dots, n.$$

With $\mathbf{x}_{\sim k} = \mathbf{x} + (x_{2k-1} - \mathbf{e}_{2k-1}^T \mathbf{x}) \mathbf{e}_{2k-1} + (x_{2k} - \mathbf{e}_{2k}^T \mathbf{x}) \mathbf{e}_{2k}$ as an arbitrary design, where the $(2k-1)$ -th and $2k$ -th components are replaced by x_{2k-1} and x_{2k} , respectively. \mathbf{e}_{2k-1} and \mathbf{e}_{2k} are unit vectors with entry in the $(2k-1)$ -th and $2k$ -th component, respectively.

Theorem 4. Let all sublevel sets as defined in (5.10) be convex:

$$\mathcal{S}_{g_c, k}(\mathbf{g}(\mathbf{x})) \text{ is convex } \forall \mathbf{x} \in \Omega_c \quad (5.11)$$

$$\forall k = 1, \dots, n.$$

Then, in order to avoid regions of bad designs within the polytope, it is sufficient to track the vertexes of the polytope.

Proof. Let Ω be a polytope and let $C_0 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \dots, \mathbf{x}_p^1\} \times \{\mathbf{x}_1^2, \mathbf{x}_2^2, \dots, \mathbf{x}_p^2\} \times \dots \times \{\mathbf{x}_1^n, \mathbf{x}_2^n, \dots, \mathbf{x}_p^n\}$ be the vertexes of the d -dimensional polytope satisfying the Solution Space constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c$. The vector $\mathbf{x}_q^k = (x_{(2k-1), q}, x_{2k, q})^T \in \mathbb{R}^2$ is the q -th vertex of the polygon in the k -th 2d-space with $q = 1, \dots, p$ and p as the number of vertexes. Furthermore, let all sublevel sets $\mathcal{S}_{g_c, k}(\mathbf{g}(\mathbf{x}))$ as defined in (5.10) be convex.

It is to show that for arbitrary $\tilde{\mathbf{x}} \in \Omega$ with $(\tilde{x}_{2k-1}, \tilde{x}_{2k}) = \sum_{q=1}^p \alpha_q \mathbf{x}_q^k, \sum_{q=1}^p \alpha_q = 1, k = 1, \dots, n$ it holds $\mathbf{g}(\tilde{\mathbf{x}}) \leq \mathbf{g}_c$.

For $\mathbf{x} \in C_0$, it is $\mathbf{x}_{q, \sim 1} = \mathbf{x} + (x_{1, q} - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1 + (x_{2, q} - \mathbf{e}_2^T \mathbf{x}) \mathbf{e}_2 \in C_0$ and $(x_{1, q}, x_{2, q}) \in \mathcal{S}_{g_c, 1}(\mathbf{g}(\mathbf{x}))$. As $(\tilde{x}_1, \tilde{x}_2) = \sum_{q=1}^p \alpha_q \mathbf{x}_q^1, \sum_{q=1}^p \alpha_q = 1$ we get $(\tilde{x}_{\sim 1}, \tilde{x}_{\sim 2}) \in \mathcal{S}_{g_c, 1}(\mathbf{g}(\mathbf{x}))$ and therefore, the inequality $\mathbf{g}(\tilde{\mathbf{x}}_{\sim 1}) \leq \mathbf{g}_c$ is valid for $\mathbf{x}_{q, \sim 1} = \mathbf{x} + (\tilde{x}_1 - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1 + (\tilde{x}_2 - \mathbf{e}_2^T \mathbf{x}) \mathbf{e}_2$. Thus, the set $C_1 = \{\tilde{\mathbf{x}}_{\sim 1} \mid \mathbf{x} + (\tilde{x}_1 - \mathbf{e}_1^T \mathbf{x}) \mathbf{e}_1 + (\tilde{x}_2 - \mathbf{e}_2^T \mathbf{x}) \mathbf{e}_2, \mathbf{x} \in C_0\}$, where any element satisfies the Solution Space constraints can be defined.

Recursively, for $\mathbf{x} \in C_{k-1}, k = 2, \dots, n$ it is $\mathbf{x}_{q, \sim k} = \mathbf{x} + (x_{(2k-1), q} - \mathbf{e}_{2k-1}^T \mathbf{x}) \mathbf{e}_{2k-1} + (x_{2k, q} - \mathbf{e}_{2k}^T \mathbf{x}) \mathbf{e}_{2k} \in C_{k-1}$ and $(x_{(2k-1), q}, x_{2k, q}) \in \mathcal{S}_{g_c, k}(\mathbf{g}(\mathbf{x}))$. Also with $(x_{(2k-1), q}, x_{2k, q}) \in \mathcal{S}_{g_c, k}(\mathbf{g}(\mathbf{x})), \mathbf{g}(\tilde{\mathbf{x}}_{\sim k}) \leq \mathbf{g}_c$ is valid for $\tilde{\mathbf{x}}_{\sim k} = \mathbf{x} + (\tilde{x}_{2k-1} - \mathbf{e}_{2k-1}^T \mathbf{x}) \mathbf{e}_{2k-1} + (\tilde{x}_{2k} - \mathbf{e}_{2k}^T \mathbf{x}) \mathbf{e}_{2k}$ and the set $C_k = \{\tilde{\mathbf{x}}_{\sim k} \mid \mathbf{x} + (\tilde{x}_{2k-1} - \mathbf{e}_{2k-1}^T \mathbf{x}) \mathbf{e}_{2k-1} + (\tilde{x}_{2k} - \mathbf{e}_{2k}^T \mathbf{x}) \mathbf{e}_{2k}, \mathbf{x} \in C_{k-1}\}$ satisfying the Solution Space constraints is defined. After a total of n iterations, we get $\tilde{\mathbf{x}}$ as the only element of C_k with $\mathbf{g}(\tilde{\mathbf{x}}) \leq \mathbf{g}_c$. □

Remark. Since the intersection of convex sets yields a convex set, Equation (5.11) can be assessed for each Solution Space constraint function $g_j, j = 1, \dots, m$ separately. However, consider that the intersection of non-convex sets may yield a convex set. This implies that even if condition (5.11) fails w.r.t. one or more Solution Space constraint function(s), it

may be satisfied if all constraint functions are considered at once. Hence, condition (5.11) assessed for each constraint function separately is sufficient but not necessary, while it is sufficient and necessary if all constraint functions are considered at once.

Theorem 5. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function, i.e. Equation (3.8) is satisfied. Then $\mathcal{S}_{g_c,k}(g(\mathbf{x}))$ is convex for all $\mathbf{x} \in \Omega_{ds}$ and w.r.t. all 2d-spaces. If this applies for all Solution Space constraint functions, condition (5.11) is satisfied.

Proof. Let g be a convex function and let $\mathbf{x}_{\sim k}$ be a design \mathbf{x} with variation by $\delta x_{2k-1}, \delta x_{2k} \geq 0$ in the $(2k-1)$ -th and $2k$ -th component. With $(x_{2k-1} + \delta x_{2k-1}, x_{2k} + \delta x_{2k}) \in \mathcal{S}_{g_c,k}(g(\mathbf{x}))$, it follows that $\mathbf{x}_{\sim k} \in \mathcal{S}_{g_c,k}(g(\mathbf{x}))$. Due to the properties of convexity, i.e. $(\tilde{x}_{2k-1}, \tilde{x}_{2k}) = \sum_{q=1}^p \alpha_q \mathbf{x}_q^k, \sum_{q=1}^p \alpha_q = 1$ and with $\mathbf{x}_{\sim k} \in \mathcal{S}_{g_c,k}(g(\mathbf{x}))$, it follows that $g(\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}_{\sim k}) \leq g_c \forall \alpha \in [0, 1]$ and hence $\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}_{\sim k} \in \mathcal{S}_{g_c,k}(g(\mathbf{x}))$. \square

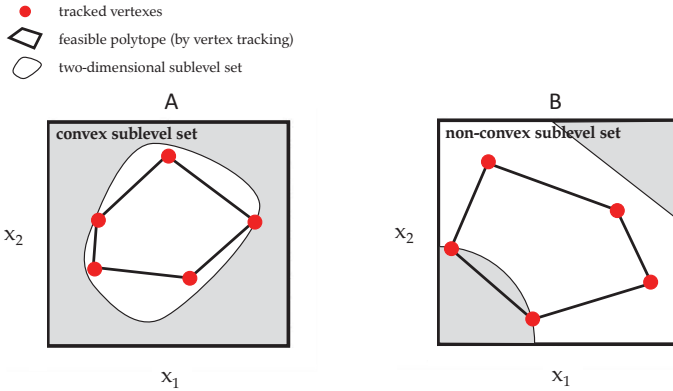


Figure 5.6: Two-dimensional sublevel sets (white areas) for design variables x_1 and x_2 of a (A) convex Solution Space Ω_c and a (B) non-convex (here concave) Solution Space. In case (A), the sublevel sets for all $\mathbf{x} \in \Omega_c$ are convex, while case (B) shows non-convex sublevel sets.

5.4.2 Problem statement

The optimization problem for maximizing the size measure of a set, described by the Cartesian product of two-dimensional polygons, where the vertexes are tracked in order to satisfy the Solution Space constraints reads

$$\begin{aligned}
 & \underset{\mathbf{x}_q^k}{\text{minimize}} && -\ln \mu(\Omega) \\
 & \text{s.t.} && \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c \quad \forall \mathbf{x} \in C_0 \subseteq \Omega_{ds}
 \end{aligned} \tag{5.12}$$

with $\mu(\Omega)$ as defined in Equation (5.3). $\mathbf{x}_q^k = (x_{(2k-1),q}, x_{2k,q})^T \in \mathbb{R}^2$ is the q -th vertex of the polygon in the k -th 2d-space with $q = 1, \dots, p$ and p as the number of vertexes. The set of d -dimensional vertexes of the polytope is denoted by C_0 and is an n -tuple of two-dimensional vertexes $C_0 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \dots, \mathbf{x}_p^1\} \times \{\mathbf{x}_1^2, \mathbf{x}_2^2, \dots, \mathbf{x}_p^2\} \times \dots \times \{\mathbf{x}_1^n, \mathbf{x}_2^n, \dots, \mathbf{x}_p^n\} \in \mathbb{R}^d$ for d even, while for d odd the last set is replaced by $\{x_d^1, x_d^1\}$, i.e. the lower and upper boundaries of the d -th design variable. The elements of the set C_0 are those designs that represent the vertexes of the d -dimensional polytope.

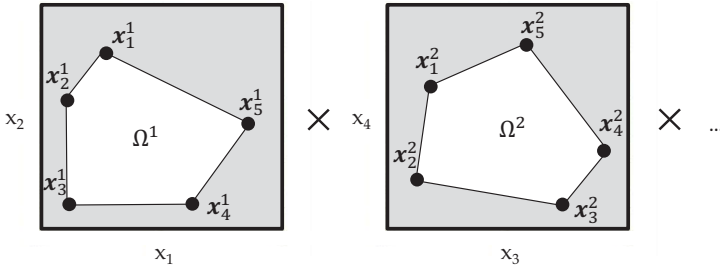


Figure 5.7: 2d-spaces each described by a polygon with p vertexes (here $p = 5$) $\mathbf{x}_q^k \in \mathbb{R}^2$, $q = 1, \dots, p$, $k = 1, \dots, n$. The number of vertexes of the d -dimensional polytope is the number of possible combinations of one out of p elements of n sets, i.e. p^n .

In the following, expressions are provided to calculate the objective function as well as the derivatives of the first and second order of problem (5.12) in dependence on the optimization parameters. d is assumed to be even. The value of the objective function for given \mathbf{x}_q^k can be obtained by computing the measures of the areas for the 2d-spaces by

$$\mu(\Omega^k) = \frac{1}{2} \sum_{q=1}^p (x_{(2k-1),q} x_{2k,(q+1)} - x_{(2k-1),(q+1)} x_{2k,q}) \tag{5.13}$$

$k = 1, \dots, n$

with $(x_{(2k-1),(p+1)}, x_{2k,(p+1)})^T = (x_{(2k-1),1}, x_{2k,1})^T$. All vertexes are in sequence with a counter-clockwise orientation w.r.t. the center of the polygon; see [4].

Derivatives of the objective function Consider the objective function as $-\ln \prod \mu(\Omega^k) = -\sum \ln \mu(\Omega^k)$ and $\frac{\partial \ln \mu(\Omega^k)}{\partial x_l^k} = 0$ for $l \neq k$, the first derivatives are

$$\begin{pmatrix} \frac{\partial(-\ln \mu(\Omega))}{\partial x_{(2k-1),q}} \\ \frac{\partial(-\ln \mu(\Omega))}{\partial x_{2k,q}} \end{pmatrix} = -\frac{1}{2\mu(\Omega^k)} \begin{pmatrix} x_{2k,(q+1)} - x_{2k,(q-1)} \\ -x_{(2k-1),(q+1)} + x_{(2k-1),(q-1)} \end{pmatrix}. \quad (5.14)$$

Note that $(q-1) = p$ holds true for $q = 1$ and $(q+1) = 1$ holds true for $q = p$.

In the following, the second derivatives are provided.

Case 1: $l = k$.

The second derivatives w.r.t. the components of the q -th and r -th vertex are

$$\begin{aligned} & \begin{pmatrix} \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{(2k-1),r} \partial x_{(2k-1),q}} \\ \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,r} \partial x_{2k,q}} \end{pmatrix} = \\ & = \frac{1}{4\mu(\Omega^k)^2} \begin{pmatrix} (x_{2k,(q+1)} - x_{2k,(q-1)}) (x_{2k,(r+1)} - x_{2k,(r-1)}) \\ (-x_{(2k-1),(q+1)} + x_{(2k-1),(q-1)}) (-x_{(2k-1),(r+1)} + x_{(2k-1),(r-1)}) \end{pmatrix} \end{aligned} \quad (5.15)$$

with $q, r = 1, \dots, p$.

For $r = q + 1$ the derivatives are

$$\begin{aligned} & \begin{pmatrix} \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,q+1} \partial x_{(2k-1),q}} \\ \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{(2k-1),q+1} \partial x_{2k,q}} \end{pmatrix} = \\ & = -\frac{1}{4\mu(\Omega^k)^2} \begin{pmatrix} (x_{2k,(q+1)} - x_{2k,(q-1)}) (x_{(2k-1),(q+2)} - x_{(2k-1),(q)}) + 2\mu(\Omega^k) \\ (-x_{(2k-1),(q+1)} + x_{(2k-1),(q-1)}) (-x_{2k,(q+2)} + x_{2k,(q)}) - 2\mu(\Omega^k) \end{pmatrix}. \end{aligned} \quad (5.16)$$

For $r = q - 1$ the derivatives are

$$\begin{aligned} & \begin{pmatrix} \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,q-1} \partial x_{(2k-1),q}} \\ \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{(2k-1),q-1} \partial x_{2k,q}} \end{pmatrix} = \\ & = -\frac{1}{4\mu(\Omega^k)^2} \begin{pmatrix} (x_{2k,(q+1)} - x_{2k,(q-1)}) (x_{(2k-1),(q)} - x_{(2k-1),(q-2)}) - 2\mu(\Omega^k) \\ (-x_{(2k-1),(q+1)} + x_{(2k-1),(q-1)}) (-x_{2k,(q)} + x_{2k,(q-2)}) + 2\mu(\Omega^k) \end{pmatrix}. \end{aligned} \quad (5.17)$$

For $r \neq q + 1$ and $r \neq q - 1$, i.e. the r -th vertex is not a neighbored vertex of the q -th

vertex, the derivatives are

$$\begin{aligned} & \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,r} \partial x_{(2k-1),q}} \right) = \\ & = -\frac{1}{4\mu(\Omega^k)^2} \begin{pmatrix} (x_{2k,(q+1)} - x_{2k,(q-1)}) (x_{(2k-1),(r+1)} - x_{(2k-1),(r-1)}) \\ (-x_{(2k-1),(q+1)} + x_{(2k-1),(q-1)}) (-x_{2k,(r+1)} + x_{2k,(r-1)}) \end{pmatrix}. \end{aligned} \quad (5.18)$$

Note that $(r-1) = p$ holds true for $r = 1$ and $(r+1) = 1$ holds true for $r = p$.

Case 2: $l \neq k$.

$$\begin{aligned} & \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{(2k-1),r} \partial x_{(2k-1),q}} \right) = \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,q+1} \partial x_{(2k-1),q}} \right) = \\ & \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,r} \partial x_{2k,q}} \right) = \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{(2k-1),q+1} \partial x_{2k,q}} \right) = \\ & \left(= \frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,q-1} \partial x_{(2k-1),q}} \right) = \left(\frac{\partial^2(-\ln \mu(\Omega))}{\partial x_{2k,r} \partial x_{(2k-1),q}} \right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned} \quad (5.19)$$

with $q, r = 1, \dots, p$.

Properties of the problem Considering Equation (5.14) to (5.19), the first and second derivatives of the objective function are positive or negative in dependence on the coordinates of the neighbored vertexes. Hence, the objective function is non-monotone and non-convex w.r.t. the optimization parameters. Without exception, the derivatives of the objective function can be calculated for all values of \mathbf{x}_p^k , which yield $\mu(\Omega^k) > 0$, and hence the objective function is twice continuous differentiable.

Figure 5.8 shows scenarios where local optima and self-intersections occur, if vertex points are moved along the gradients, i.e. if a gradient-based optimizer is used. The components of the gradient of $\ln \mu(\Omega)$, computed by Equation (5.14), are shown as red arrows. Figure 5.8 (A) and (B) show polygons with the position of the vertexes (black dots) as well as the globally optimal polygon (black dashed lines). Moving the vertexes along the gradients and hence increasing the size measure of the area $\mu(\Omega)$ will lead to a polygon, violating the constraints (gray area). Hence, the polygons (black bold lines) are local optima. Figure 5.8 (C) shows a scenario where moving vertex point 2 along its

gradient leads to self-intersection, however, self-intersecting polygons are not desired.

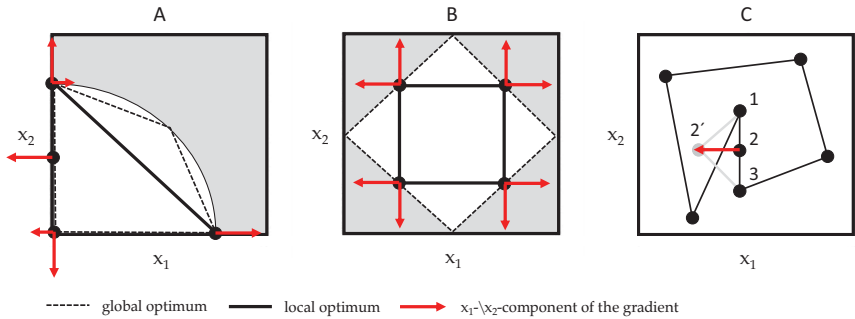


Figure 5.8: Two-dimensional example with the region of good designs as white area: (A) and (B) show a polygon (black bold lines) with four vertexes as well as the components of the gradient in x_1 and x_2 direction (red arrows). Additionally, the global optimum is shown (dotted lines). (C) shows a polygon and the gradient of a vertex point potentially causing self-intersections if the vertex point is moved along the direction of the gradient.

5.4.3 Implementation

Optimization algorithm Since the derivatives of the objective function are available, the gradient-based interior-point optimizer [15, 16, 71] in the MATLAB[®] environment, called by the command `fmincon()`, is used. Depending on whether the Solution Space constraints are given as parametric or oracle functions, the derivatives can be provided or must be computed by e.g. finite-differences. For the validity of vertex tracking in conjunction with polytopes, condition (5.11) must be checked.

Self-intersecting polygons Polygons with self-intersecting edges are not desired (*remark*: Equation (5.13) does not hold for such types of polygons). In order to avoid this during optimization, self-intersections must be detected. Therefore, possible interactions of non-neighbored edges are calculated (the intersection of non-neighbored edges of the polygon indicate a self-intersection) and the objective function value is set to 'nan' in case of self-intersections. Hence, the optimizer will reject the step.

Determination of initial values The initial values of \mathbf{x}_q^k for optimization problem (5.12) must be selected such that the two-dimensional polygons obtained show no self-intersections

and the vertex points of the polygon are oriented counter-clockwise w.r.t. the center of the polygon. The latter is necessary to apply Equation (5.13).

Based on the fact that the optimization problem to be solved shows local optima, it is advantageous to generate the initial values randomly and repeating the experiment. In order to start with a feasible initial guess, p vertex points are generated randomly within $[x_{2k-1}^l, x_{2k-1}^u] \times [x_{2k}^l, x_{2k}^u]$; $[x_{2k-1}^l, x_{2k-1}^u]$ and $[x_{2k}^l, x_{2k}^u]$ as the intervals of a solution box, i.e. a box that contains good designs only. Subsequently, the vertexes are sequenced in a counter-clockwise orientation w.r.t. the center of the polygon.

5.4.4 Numerical results

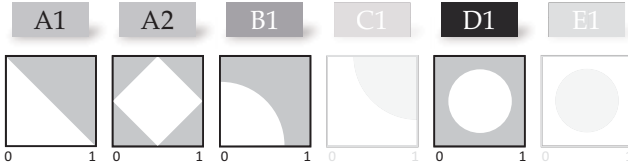


Figure 5.9: Overview of the test problems for $d = 2$. Only the convex problems are considered, due to the requirements of the approach.

In order to evaluate the effort for solving problem (5.12) in dependence on the number of dimensions, the number of constraints and the number of vertexes of the two-dimensional polygons, the approach presented is applied to test problems $A1$, $A2$, $B1$ and $D1$. All test problems are convex and hence vertex tracking can be applied. For all test problems considered, the design space is $\Omega_{\text{ds}} = [0, 1]^d$. The initial values for test problem $A1$ and $B1$ are obtained by generating four and six vertex points within the solution box $\Omega = [0.01, 0.11]^d$ for each 2d-space separately, i.e. within $[0.01, 0.11] \times [0.01, 0.11]$. For test problems $A2$ and $D1$ the initial values are randomly generated vertex points within $[0.45, 0.55] \times [0.45, 0.55]$ for each 2d-space. Each run is repeated ten times with randomly selected initial values and the best result in terms of a minimum discrepancy to the analytic solution is shown in Table 5.1. For the relative error, the size measure obtained by numerical optimization $\mu(\Omega)_{\text{nn}}$ and the optimal size measure obtained by analytic calculus $\mu(\Omega)_{\text{an}}$ is considered. The relative error is defined as $\frac{\mu(\Omega)_{\text{nn}} - \mu(\Omega)_{\text{an}}}{\mu(\Omega)_{\text{an}}}$. Table 3.2 shows the formula for the optimal size measures, which depends on the number of vertexes p in case of $B1$ and $D1$. Based on the fact that the number of optimization constraints scales with p^n , solving optimization problem (5.12) is not practicable for $d = 50$ and hence is not considered here. Technical details about the computer and software used are provided in the Appendix.

For values with an asterisk (*), all of the ten runs terminated, because the step size

Table 5.1: Numerical effort for two linear ($A1$, $A2$) and two nonlinear but convex ($B1$, $D1$) test problems. While the number of constraints are kept constant (for test problem $A1$, $B1$, $D1$ $m = 1$ and for $A2$ $m = 4$) the number of dimensions is increased. The examinations are executed with $p = 4$ (row 1-4) and $p = 6$ (row 5-8), i.e. with polygons with four and six vertexes, respectively. The values are associated with the following number of dimensions $d = 2, 4, 10$.

	no. iterations	no. fun. eval.	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
$A1$	14, 18, 21	19, 22, 26	0.31, 0.41, 1.15	-8e-5, -1e-4, -3e-4
$A2$	10, 14, 15	11, 18, 17	0.15, 0.36, 4.80	-2e-5, -3e-5, -3e-4
$B1$	18, 21, 122	19, 22, 148	0.35, 1.40, 393.65	-1e-5, -3e-5, -7e-3
$D1$	22, 31, 230*	25, 35, 525*	0.60, 0.52, 373.97*	-8e-5, -3e-5, -4.11*
$A1$	19*, 19*, 19*	86*, 88*, 112*	1.41*, 1.88*, 8.57*	-3*, -22*, -6*
$A2$	25, 29, 23	43, 39, 28	1.23, 2.30, 142.61	-6e-4, -5e-5, -1e-2
$B1$	31, 307*, X	32, 325*, X	1.45, 24.88*, X	-1e-5, -6*, X
$D1$	22, 39, 211	23, 56, 596	0.89, 1.99, 1871.29	-5e-5, -2e-4, -2e-1

* sub-optimal results, critical step size reached

X maximum number of iterations reached

reached a specified threshold of $1e-10$. Figure 5.10 shows polygons generated for each iteration of the optimization process for test problem $D1$ with $d = 4$ and $p = 4$. One polygon shows self-intersection, which can be observed more frequently when the number of dimensions and the number of vertexes is increased. As a consequence, the optimizer rejects the step and computes a new step, which leads to a high number of overall function evaluations or termination of the optimization due to a step size less than the specified threshold. Runs without an asterisk reached a final first-order optimality measure less than the critical threshold of $1e-6$, and a final maximum constraint violation less than the critical threshold of $1e-6$. For test problem $B1$ with $d = 10$ and $p = 6$, all of the runs exceeded a maximum number of iterations of 400 (marked by an X).

Discussion The results in Table 5.1 show that the linear test problems $A1$ and $A2$ with $p = 4$ can be solved efficiently in terms of a low CPU time of a few seconds and accurately in terms of a relative error between the numerical results and the analytic solution of less than $1e-3$. However, if the number of vertexes is increased to $p = 6$, the optimization problem gets more challenging, and no optimal solution is found in case of $A1$ (marked by an asterisk (*)), whereas the computational effort is increased in case of $A2$. Note that the optimal result of $A1$ are triangles within each 2d-design space, and hence, in the case of six vertexes for each polygon, three vertexes lie on the edges of the triangle with undetermined position. In the case of test problem $A2$, the optimal shape of the polygons are rectangles,

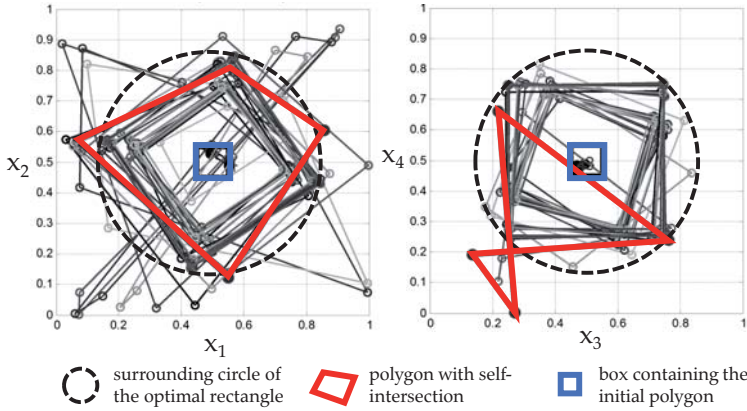


Figure 5.10: Polygons generated by the optimizer in each iteration for test problem $D1$ with $d = 4$ and $p = 4$.

and hence for $p = 6$ two vertex points in each 2d-design space lie on the edges of the polygon. Depending on the initial position of the vertex points, the optimizer is not able to reach the optimal solution due to local minima. Three examples are depicted in Figure 5.8. Additionally, self-intersections may occur, which are treated as infeasible as discussed in Section 5.4.3, causing the optimizer to recalculate the step, and often termination due to a step size beyond the critical threshold occurs. The results of test problem $B1$ for $p = 4$ show that the problem can be solved efficiently for $d = 2$ and $d = 4$, but requires a CPU time of a few minutes in case of $d = 10$. The latter is due to the fact that the number of optimization constraints scales with p^n and self-intersections occur more frequently in presence of a large number of dimensions. Moreover, the optimizer terminated due to a step size less than $1e-10$ for test problem $D1$ with $d = 10$. Increasing the number of vertices to $p = 6$ shows that in the case of $B1$ the optimizer is not able to find the optimal solution for $d = 4$ and $d = 10$. For problem $D1$ with $d = 10$ the CPU time increased significantly compared to $D1$ with $d = 4$.

In summary, considering more vertex points p and more dimensions d increases the numerical effort significantly, due to an increasing number of optimization constraints scaling with p^n and an increasing occurrence of self-intersections. This often leads to a termination of the optimizer due to step sizes beyond the critical threshold. Problems where the optimal shape of each polygon shows symmetry reveal less issues in terms of a termination of the optimizer due to small step sizes, high CPU times and inaccurate

results.

5.5 Decomposing Solution Space constraints

In this section the idea of decomposing d -dimensional Solution Space constraints into a set of two-dimensional Solution Space constraints is used to reformulate the general problem statement (5.1), such that it can be solved by any standard optimization algorithm. Firstly, linear Solution Space constraints are assumed. The following expressions are in accordance with Erschen et al. [26]. At the end of this section the approach is generalized to nonlinear problems; see also Daub [19].

5.5.1 Problem statement

The problem statement for seeking optimal 2d-spaces based on a two-dimensional decomposition of Solution Space constraints reads

$$\begin{aligned} & \underset{g_{c,j}^k}{\text{minimize}} && -\ln \mu(\Omega) \\ & \text{s.t.} && \sum_{k=1}^n g_{c,j}^k = g_{c,j} \quad \forall j = 1, \dots, m \\ & && \Omega := \Omega(g_{c,1}^1, \dots, g_{c,m}^n) \subseteq \Omega_{\text{ds}}. \end{aligned} \quad (5.20)$$

With $\mu(\Omega)$ as defined in Equation (5.3), which depends on the optimization parameters, i.e. the thresholds of the two-dimensional inequalities denoted by $g_{c,j}^k$. The equality constraint is equal to Equation (5.9) and guarantees that a design with all pairs of design variables within their associated 2d-spaces, satisfies the overall requirements on the system. In the case where the Solution Space constraints are linear functions, the negative logarithm of problem statement (5.20) yields a convex objective function; see proof below. In conjunction with the linear optimization constraints, this leads to a convex optimization problem. This implies that a local minimum is also the global minimum, and that effective computational methods are available; see [13]. Additionally, as already mentioned in Section 4.5.1, the logarithm acts as a barrier for $\mu(\Omega) \rightarrow 0$ and 2d-spaces with small size measures are avoided.

In the following, expressions are provided to compute the objective function for given $g_{c,j}^k$ as well as the derivatives of the first and second order of problem (5.20). For simplicity of the presentation, d is assumed to be even. In the linear case, the Solution Space constraints are given in the form $\mathbf{G}\mathbf{x} \leq \mathbf{g}_c$. The coefficients of the linear inequalities are given by $\mathbf{G} \in \mathbb{R}^{m \times d}$ with elements G_{ji} , $i = 1, \dots, d$, $j = 1, \dots, m$, and the thresholds are denoted by $\mathbf{g}_c \in \mathbb{R}^m$. In accordance with Equation (5.7), the linear inequalities are decomposed into a set of two-dimensional inequalities $G_{j,(2k-1)}x_{2k-1} + G_{j,2k}x_{2k} \leq g_{c,j}^k$, $j = 1, \dots, m$, $k = 1, \dots, n$. The equality $G_{j,(2k-1)}x_{2k-1} + G_{j,2k}x_{2k} = g_{c,j}^k$ is called a **boundary**.

The associated **normal vector** is

$$\mathbf{n}_j^k = \frac{(G_{j,(2k-1)}, G_{j,2k})^T}{\|(G_{j,(2k-1)}, G_{j,2k})\|}. \quad (5.21)$$

Each set of design variables specified by a 2d-space is enclosed by a polygon. A boundary is active if it is part of the polygon, i.e. evokes one of the **edges**. Each active boundary has two neighbors, i.e. boundaries that intersect in the two associated **vertexes**; see Figure 5.11.

Computing the size of a 2d-space, taken from [26] For computing the measure of the area for each 2d-space as defined in (5.4), the vertexes of each two-dimensional polygon must be computed; see Figure 5.11. Therefore, $\binom{m+4}{2} = \frac{(m+4)^2 - (m+4)}{2}$ systems of two two-dimensional linear equations must be solved in order to derive all possible intersections of the boundaries of the constraints and the 2d-design space boundaries, i.e. $x_{2k-1} = x_{2k-1}^{\text{lb}}, x_{2k-1} = x_{2k-1}^{\text{ub}}, x_{2k} = x_{2k}^{\text{lb}}, x_{2k} = x_{2k}^{\text{ub}}$. An intersection point is a vertex of the polygon, if it satisfies all m two-dimensional inequalities for the associated 2d-space and lies within the associated 2d-design space Ω_{ds}^k . The q -th vertex in the k -th 2d-space is denoted by the vector $(x_{(2k-1),q}, x_{2k,q})^T \in \mathbb{R}^2$. Hence, $\mu(\Omega^k)$ can be calculated by Equation (5.13), which is

$$\mu(\Omega^k) = \frac{1}{2} \sum_{q=1}^p (x_{(2k-1),q} x_{2k,(q+1)} - x_{(2k-1),(q+1)} x_{2k,q})$$

with $k = 1, \dots, n$ and p as the *number of vertexes*. Note that it holds $(x_{(2k-1),(p+1)}, x_{2k,(p+1)})^T = (x_{(2k-1),1}, x_{2k,1})^T$. All vertexes are in sequence with a counter-clockwise orientation w.r.t. the center of the polygon; see [4].

Derivatives of the objective function, taken from [26] The first derivatives of the objective function are

$$\frac{\partial(-\ln \mu(\Omega))}{\partial g_{c,j}^k} = -\frac{\partial \mu(\Omega^k)}{\partial g_{c,j}^k} \frac{1}{\mu(\Omega^k)} \quad (5.22)$$

with $j = 1, \dots, m$ and $k = 1, \dots, n$. Substituting the derivative terms on the right hand side yields the following expressions.

Case 1: $G_{j,(2k-1)}^2 + G_{j,2k}^2 \neq 0$ and *boundary of constraint j is active* (both conditions must apply).

$$\frac{\partial(-\ln \mu(\Omega))}{\partial g_{c,j}^k} = -\frac{w_{j,0}^k}{\sqrt{G_{j,(2k-1)}^2 + G_{j,2k}^2}} \frac{1}{\mu(\Omega^k)}. \quad (5.23)$$

$w_{j,0}^k$ is the **edge length** of the polygon of the k -th 2d-space, evoked by the j -th Solution Space constraint. It is calculated as the distance between the intersection points of the

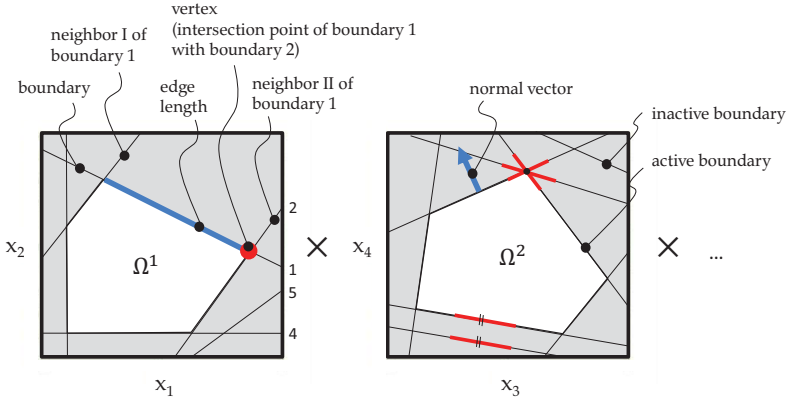


Figure 5.11: 2d-spaces constrained by linear inequalities, with the edge length $w_{1,0}^1$ of the boundary $G_{1,1}x_1 + G_{1,2}x_2 = g_{c,1}^1$ (blue line), and the vertex $\mathbf{x}_{1,II}^1 \in \mathbb{R}^2$ (red dot) as an intersection point of the first constraint with its second neighbor. In the second 2d-space a normal vector \mathbf{n}_j^k of a boundary is depicted. Highlighted by red bold lines are cases where three boundaries intersect in one vertex and where two boundaries have identical normal vectors, i.e. are parallel.

j -th constraint with its two neighbors, denoted by $\mathbf{x}_{j,I}^k \in \mathbb{R}^2$ and $\mathbf{x}_{j,II}^k \in \mathbb{R}^2$.

$$w_{j,0}^k = \begin{cases} \|\mathbf{x}_{j,I}^k - \mathbf{x}_{j,II}^k\| & \text{if constraint } j \text{ has two neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (5.24)$$

Case 2: Either $G_{j,(2k-1)}^2 + G_{j,2k}^2 = 0$, or *boundary of constraint j is not active*.

$$\frac{\partial(-\ln \mu(\Omega))}{\partial g_{c,j}^k} = 0. \quad (5.25)$$

The Hessian of the objective function reads

$$H_{rsjk} = \frac{\partial^2(-\ln \mu(\Omega))}{\partial g_{c,r}^s \partial g_{c,j}^k} = -\frac{1}{\mu(\Omega^k)^2} \left(\mu(\Omega^k) \frac{\partial^2 \mu(\Omega^k)}{\partial g_{c,r}^s \partial g_{c,j}^k} - \frac{\partial \mu(\Omega^k)}{\partial g_{c,j}^k} \frac{\partial \mu(\Omega^k)}{\partial g_{c,r}^s} \right) \quad (5.26)$$

with $j, r = 1, \dots, m$ and $k, s = 1, \dots, n$. The following expressions are obtained by substituting the derivative terms in Equation (5.26); for details see [26].

Case 1: $G_{j,(2k-1)}^2 + G_{j,2k}^2 \neq 0$, $G_{r,(2k-1)}^2 + G_{r,2k}^2 \neq 0$, $s = k$ and *both, boundary of constraint j and boundary of constraint r are active* (all conditions must apply).

$$H_{rsjk} = \begin{cases} \frac{\left(-\frac{(\mathbf{n}_{j,I}^k)^T \mathbf{n}_j^k}{\sqrt{1 - ((\mathbf{n}_{j,I}^k)^T \mathbf{n}_j^k)^2}} - \frac{(\mathbf{n}_{j,II}^k)^T \mathbf{n}_j^k}{\sqrt{1 - ((\mathbf{n}_{j,II}^k)^T \mathbf{n}_j^k)^2}} \right) \mu(\Omega^k) - (w_{j,0}^k)^2}{\mu(\Omega^k)^2 (G_{j,(2k-1)}^2 + G_{j,2k}^2)} & \text{for } r = j \\ \frac{\mu(\Omega^k) \frac{1}{\sqrt{1 - ((\mathbf{n}_{j,r}^k)^T \mathbf{n}_j^k)^2}} - w_{j,0}^k w_{r,0}^k}{\mu(\Omega^k)^2 \sqrt{G_{j,(2k-1)}^2 + G_{j,2k}^2} \sqrt{G_{r,(2k-1)}^2 + G_{r,2k}^2}} & \text{for } r \neq j \text{ but intersection with } j \\ \frac{-w_{j,0}^k w_{r,0}^k}{\mu(\Omega^k)^2 \sqrt{G_{j,(2k-1)}^2 + G_{j,2k}^2} \sqrt{G_{r,(2k-1)}^2 + G_{r,2k}^2}} & \text{for } r \neq j \text{ no intersection with } j \end{cases} \quad (5.27)$$

$\mathbf{n}_{j,I}^k$ and $\mathbf{n}_{j,II}^k$ are the normal vectors of the neighbors of constraint j . $\mathbf{n}_{j,r}^k$ is the normal vector of the r -th constraint, whose boundary has an intersection with the boundary of the j -th constraint, i.e. either $\mathbf{n}_{j,I}^k$ or $\mathbf{n}_{j,II}^k$.

Case 2: $G_{j,(2k-1)}^2 + G_{j,2k}^2 = 0$, $G_{r,(2k-1)}^2 + G_{r,2k}^2 = 0$, $s \neq k$ or *boundary of constraint j or boundary of constraint r are not active* (at least one of the conditions must apply).

$$H_{rsjk} = 0. \quad (5.28)$$

Properties of the problem, taken from [26] The enclosed area of the k -th 2d-space $\mu(\Omega^k)$ increases monotonously with increasing $g_{c,j}^k$. This can be explained by the fact that a relaxation of the restriction, i.e. increasing $g_{c,j}^k$, can only yield more or equal good designs. A change in $g_{c,j}^k$ affects only $\mu(\Omega^k)$, while the areas of the other $n - 1$ 2d-spaces are not affected. This means that $\mu(\Omega)$ depends monotonously on $g_{c,j}^k$, and hence the objective function of optimization problem (5.20) depends monotonously on $g_{c,j}^k$.

To show the convexity of the objective function, refer to Boyd [13]. Boyd states that the volume of a polyhedron (as it is $\mu(\Omega^k)$), where the polyhedron is defined as a set $\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ (the inequalities hold component-by-component for each entry of the vector-valued function), is a log-concave function of \mathbf{b} , which is equivalent to $g_{c,j}^k$ for 2d-spaces. A function f is called log-concave, if it holds $f(\mathbf{x}) > 0$ with \mathbf{x} within the domain of f and if the logarithm of f is concave. (*remark:* Boyd: "It is convenient to allow f to take on the value zero", pg. 104). Consider the logarithm of $\mu(\Omega)$ as $\ln \prod \mu(\Omega^k) = \sum \ln \mu(\Omega^k)$ with $\mu(\Omega^k)$ as a log-concave function w.r.t. $g_{c,j}^k$. Since the sum of concave functions is concave, $\mu(\Omega)$ is a log-concave function of $g_{c,j}^k$. Consequently, the negative logarithm of $\mu(\Omega)$ is convex w.r.t. $g_{c,j}^k$.

The objective function is not differentiable at points where two boundaries with equal normal vectors as defined in (5.21) coincide; see Figure 5.12 (top right and bottom, dashed

lines, for $\Delta g_{c,j}^k = 0$). For problems satisfying the following condition, i.e. all normal vectors in one 2d-space are different, the objective function is once continuous differentiable. The necessary condition reads

$$\mathbf{n}_j^k \neq \mathbf{n}_r^k \quad (5.29)$$

$$\forall j, r = 1, \dots, m, j \neq r.$$

Furthermore, the first derivative is not differentiable at points where at least three boundaries intersect in one vertex of the polygon, i.e. at those points where the number of vertexes of the polygon changes with changes in $g_{c,j}^k$; see Figure 5.12 (top left and bottom, red solid line, for $\Delta g_{c,j}^k = -0.05$ and $\Delta g_{c,j}^k = 0.05$).

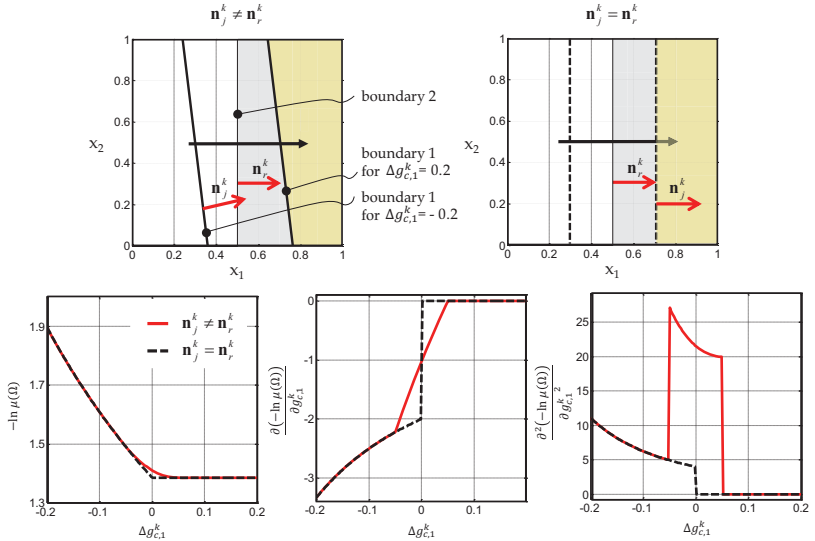


Figure 5.12: Two cases where the threshold of the two-dimensional inequality of constraint 1, $g_{c,1}^k$, is increased and hence, the boundary of constraints 1 moves from left to right. Top left: The normal vectors of the boundaries of constraint 1 and constraint 2 are different. Top right: The normal vectors are identical. Bottom: The value of the objective function of problem (5.20) as well as the first and second derivatives.

5.5.2 Implementation

Optimization algorithm Based on the fact that the derivatives of the objective function and constraints are available and the objective function is continuous and convex, a

gradient-based optimization algorithm is preferred. A Newton's method based algorithm, namely the interior-point implementation in the MATLAB® environment [22, 38, 77] called by the command $fmincon()$, is used. In the case of nonlinear Solution Space constraints, a non-gradient-based optimization algorithm is used; see paragraph below.

Bounds of the optimization parameters Test runs revealed that the performance in terms of convergence was improved by setting additional bounds to the optimization parameters:

$$g_{c,j}^{k,lb} \leq g_{c,j}^k \leq g_{c,j}^{k,ub}. \quad (5.30)$$

The bounds are calculated as

$$g_{c,j}^{k,lb} = \min\{G_{j,(2k-1)x_{2k-1}} + G_{j,2kx_{2k}} \mid x_{2k-1}, x_{2k} \in \Omega_{ds}^k\} \quad (5.31)$$

and

$$g_{c,j}^{k,ub} = \max\{G_{j,(2k-1)x_{2k-1}} + G_{j,2kx_{2k}} \mid x_{2k-1}, x_{2k} \in \Omega_{ds}^k\}. \quad (5.32)$$

Note that these bounds are not necessary from a mathematical point of view, since the optimum denoted by (*) lies within the bounds, i.e. $g_{c,j}^{k*} \in [g_{c,j}^{k,lb}, g_{c,j}^{k,ub}]$. This can be explained by the following: $\mu(\Omega^k)$ assumes its minimum value if $g_{c,j}^k = g_{c,j}^{k,lb}$ (no combination of associated design variables in the k -th 2d-space is allowed, i.e. $\mu(\Omega^k) = 0$) and assumes its maximum value if $g_{c,j}^k = g_{c,j}^{k,ub}$ (all combinations of associated design variables in the k -th 2d-space are allowed, assuming $m = 1$). Cases with $g_{c,j}^k < g_{c,j}^{k,lb}$ and $g_{c,j}^k > g_{c,j}^{k,ub}$ do not affect $\mu(\Omega^k)$, and hence do not affect $\mu(\Omega)$.

Determination of initial values The initial values of problem (5.20) for $g_{c,j}^k$ must be selected such that $\mu(\Omega^k) > 0$, $\forall k = 1, \dots, n$ and such that the equality constraints $\sum_{k=1}^n g_{c,j}^k = g_{c,j}$, $\forall j = 1, \dots, m$ are satisfied. The former is crucial due to the fact that for $\mu(\Omega^k) = 0$, the value of the objective function is not defined, and the determination of a search direction by the optimizer is impossible.

Strategy 1: The determination of suitable initial values can be achieved by taking any solution box, i.e. a box that contains good designs only, and computing the values of $g_{c,j}^k$ as the maximum values of $G_{j,(2k-1)x_{2k-1}} + G_{j,2kx_{2k}}$ within that box, i.e. $\max\{G_{j,(2k-1)x_{2k-1}} + G_{j,2kx_{2k}} \mid x_{2k-1}, x_{2k} \in [x_{2k-1}^1, x_{2k-1}^n] \times [x_{2k}^1, x_{2k}^n]\}$. This can be achieved by applying interval arithmetic; see Section 4.2.2. If necessary, the values obtained must be increased to satisfy the equality constraint $\sum_{k=1}^n g_{c,j}^k = g_{c,j}$, $\forall j = 1, \dots, m$ of optimization problem (5.20). This is accomplished by starting in the first 2d-space and increasing the values of $g_{c,j}^k$, until either the equality constraint is satisfied or until the value exceeds the bounds specified in constraint (5.30). In the latter case, the procedure is repeated in the next 2d-design space until the equality constraint is satisfied.

Strategy 2: An alternative strategy for the determination of initial values, which yield $\mu(\Omega^k) > 0$, $\forall k = 1, \dots, n$ and satisfy $\sum_{c,j=1}^n g_{c,j}^k = g_{c,j}$, $\forall j = 1, \dots, m$, is to compute the value of the Solution Space constraint functions $\mathbf{g}(\mathbf{x}) = \mathbf{G}\mathbf{x}$ for one single permissible design \mathbf{x}^* . The design can be found by solving e.g. the following optimization problem: minimize α s.t. $\mathbf{g}(\mathbf{x}) \leq \mathbf{g}_c + \alpha$. For each pair of design variables associated with a particular 2d-space, the values of $G_{j,(2k-1)}x_{2k-1}^* + G_{j,2k}x_{2k}^*$ are calculated. In order to satisfy the equality constraint, the gap between the obtained $\mathbf{g}(\mathbf{x}^*)$ and \mathbf{g}_c is equally distributed to all values obtained. The equation is

$$g_{c,j,0}^k = \frac{g_{c,j} - g_j(\mathbf{x}^*)}{n} + (G_{j,(2k-1)}x_{2k-1}^* + G_{j,2k}x_{2k}^*). \quad (5.33)$$

In case of violation of (5.30) the values of $g_{c,j}^k$ must be adjusted appropriately.

Setting additional constraints Test runs showed that the optimizer often aborts for cases with $G_{j,(2k-1)} = G_{j,2k} = 0$. The associated two-dimensional inequality reads $0 \leq g_{c,j}^k$. This means that all combinations of design variables within the associated design space are feasible as long as $g_{c,j}^k \geq 0$. For $g_{c,j}^k < 0$ the 2d-space becomes infeasible. To avoid the abrupt change in the objective function, an additional equality constraint is introduced, which is $g_{c,j}^k = 0$ ¹ for all j, k with $G_{j,(2k-1)} = G_{j,2k} = 0$.

Nonlinear Solution Space constraints In case of nonlinear Solution Space constraints, the measures of the areas of the 2d-spaces and consequently the objective function cannot be computed via the vertexes of two-dimensional polygons. For nonlinear functions, the areas assume arbitrary shapes. One approach to compute $\mu(\Omega^k)$ is to estimate the measures of the areas by sample points within each 2d-design space, distributed on a dense grid. The sample points are categorized as good or bad depending on whether they satisfy all two-dimensional inequalities of the associated 2d-space or not. The estimated (normalized) size measure of the area for a uniform sampling is

$$\hat{\mu}(\Omega^k) = \frac{\tilde{N}_g}{\tilde{N}} \quad (5.34)$$

with \tilde{N}_g as the number of good sample points out of \tilde{N} sample points. For a sufficient estimation \tilde{N} must be of order $1e6$. This requires that the evaluation of $g_j^k(x_{2k-1}, x_{2k})$ is computationally inexpensive. Since the derivatives are not available and finite-differences derived from the estimate of $\mu(\Omega^k)$ showed insufficient results even for a very large number of sample points, a non-gradient-based (but deterministic) optimizer is used, namely

¹Due to computational inaccuracies, it is advisable to select $g_{c,j}^k$ as a positive value close to zero, e.g. $g_{c,j}^k = 1e-4$.

the pattern search algorithm in the MATLAB® environment, called by the command `patternsearch()`. For more information about pattern search algorithms see e.g. [7, 8], the fundamentals are explained in Appendix A.2.

5.5.3 Numerical results

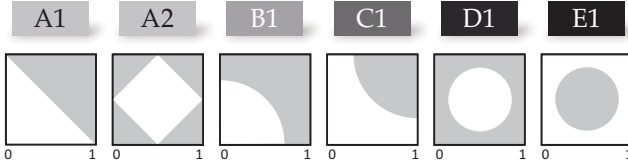


Figure 5.13: Overview of the test problems for $d = 2$. All linear and nonlinear problems are considered.

For the evaluation of the numerical effort for solving optimization problem (5.20) test problems *A1*, *A2*, *B1*, *C1*, *D1* and *E1* are considered; see Table 3.1. The design space for all examples is given by $\Omega_{\text{ds}} = [0, 1]^d$. For the determination of the initial values of $g_{c,j}^k$ Strategy 1, proposed in Section 5.5.2, is applied: The initial solution box for test problems *A1*, *B1*, *C1* and *E1* is $\Omega = [0.01, 0.11]^d$. For test problems *A2* and *D1* it is given by $\Omega = [0.45, 0.55]^d$. Table 5.2 shows the number of iterations, the number of function evaluations, the CPU time as well as the relative error of the obtained results compared to the analytic solution. The relative error is defined as $\frac{\mu(\Omega)_{\text{nu}} - \mu(\Omega)_{\text{an}}}{\mu(\Omega)_{\text{an}}}$, with $\mu(\Omega)_{\text{nu}}$ as the size measure obtained by solving problem (5.20) numerically and $\mu(\Omega)_{\text{an}}$ as the analytic solution, computed by the formula provided in Table 3.2. Note that for $d = 2$ there is no degree of freedom, i.e. a feasible initial guess with all constraints satisfied is globally optimal. Hence, all cases with $d = 2$ are not considered in Table 5.2. Technical details about the computer and software used are provided in the Appendix.

For the linear test problems, the gradient-based interior-point algorithm called by `fmincon()` is used, while the nonlinear test problems are solved by a non-gradient-based optimizer, called by the command `patternsearch()` in the MATLAB® environment. For all optimization runs executed by `fmincon()` the final first-order optimality measure and the final maximum constraint violation are less than the critical threshold of $1e-6$. For more information about the first-order optimality measure see Appendix A.1. All optimization runs executed by `patternsearch()` terminated due to a final mesh size less than the critical threshold of $1e-6$. However, the threshold for the constraint violation of $1e-6$ cannot be satisfied by the optimizer, but does not exceed $1e-3$ for all results obtained.

Table 5.2: Numerical effort for the computation of the optimal 2d-spaces for the linear test problems $A1$ and $A2$ as well as for the quadratic test problems $B1$, $C1$, $D1$ and $E2$. While the number of constraints are kept constant (for test problem $A1$ $m = 1$ and for $A2$ $m = 4$), the number of dimensions is increased. The values are associated with the following number of dimensions $d = 4, 10, 50$.

	no. iterations	no. fun. eval.	CPU time [sec]	rel. error of $\mu(\Omega)$ [%]
$A1$	8, 8, 15	13, 9, 16	0.21, 0.44, 1.49	2e-14, 4e-14, 2e-12
$A2$	9, 11, 12	11, 12, 17	0.30, 2.02, 2.29	2e-14, 4e-14, 2e-12
$B1$	32, 58, 286	127, 469, 7462	0.65, 1.91, 44.63	1e-1, -3.e-1, -3
$C1$	26, 56, 66	109, 447, 1607	0.33, 0.82, 16.63	5.e-1, 1, 2
$D1$	36, 48, 450	129, 381, 10378	0.39, 2.12, 110.82	-3, -7, -34
$E1$	32, 50, 60	117, 391, 1550	0.67, 1.23, 11.73	3, 7, 14

Discussion Considering the results, Table 5.2 shows that the number of iterations as well as the CPU time slightly increases with an increasing number of dimensions for the linear test problems $A1$ and $A2$. The number of function evaluations is larger than the number of iterations, which implies that the algorithm computes additional steps in order to adapt the step size appropriately. The relative error in all cases is negligible.

Based on the fact that a non-gradient-based optimizer is used in the case of the nonlinear test problems $B1$ to $E1$, the number of iterations and particularly function evaluations is significantly higher compared to the linear case, where a gradient-based optimizer is used. Nevertheless, with the exception of cases with $d = 50$, the CPU time is approximately equal to that of the linear test problems. This can be explained by the fact that the computational effort for the determination of a step in each iteration is negligible for the pattern search algorithm [7, 8] compared to the interior-point implementation where LDL factorization is accomplished; see Appendix A.2. Additionally, the evaluation of the Solution Space constraint functions is very inexpensive as they are provided as parametric functions. For $d = 50$ the number of iterations and the CPU time increases significantly, particularly for the convex test problems $B1$ and $D1$. The relative error is remarkably high for most of the results with a size measure up to 34% smaller compared to the analytic solution. As mentioned in the paragraph above, the optimizer is not able to satisfy the constraint to a precision of 1e-6 but to a precision of 1e-3 only.

In summary, both the linear and nonlinear test problems are solved within a low CPU time, which exceeds a few seconds only in case of $d = 50$. While the numerical results obtained by `fmincon()` are precise in terms of the constraint violation and the discrepancy to the analytic solution, the results obtained by `patternsearch()` are suboptimal in most of the cases.

6 — Comparison of the approaches

In the previous chapters, numerical approaches for the computation of one- and two-dimensional representations of high-dimensional Solution Spaces were introduced and reviewed. In the following, the results of Chapter 4 and 5 are summarized and compared. Firstly, the underlying optimization problems for the computation of box-shaped Solution Spaces and 2d-spaces are compared in terms of the properties of the optimization problems. Secondly, the numerical effort in dependence on the number of dimensions and the number of Solution Space constraints is analyzed and discussed. A comparison between computing a box-shaped Solution Space by a gradient-based optimizer in conjunction with vertex tracking and by the stochastic Solution Space algorithm as well as a comparison between computing box-shaped Solution Spaces and 2d-spaces is provided.

Table 6.1: Comparison of the different approaches ($V1$, $W1$, $W2$) regarding the properties of the objective functions and optimization constraints, the number of optimization parameters N , the number of linear optimization constraints P , the number of nonlinear optimization constraints Q and the number of derivatives of nonlinear optimization constraints D . d is the number of design variables (which are assumed to be even), m is the number of Solution Space constraints, p is the number of vertexes of a 2-dimensional polygon and n is the number of 2d-spaces ($n = d/2$ for d even). $2N$ is the number of bounds of the optimization parameters, i.e. linear inequalities.

approach	obj. fun.	opt. con.	N	P	Q	D
$V1$, lin. (4.12)	convex	convex	$2d$	$m + d + 2N$	–	–
$W1$, lin. (5.12)	n. convex	convex	$2pn$	$mp^n + 2N$	–	–
$W2$, lin. (5.20)	convex	convex	mn	$m + 2N$	–	–
$V1$, n.lin. (4.24)	convex	conv./n. conv.	$2d$	$d + 2N$	$[m, m2^d]$	$[mN, m2^dN]$
$W1$, n.lin. (5.12)	n. convex	conv./n. conv.	$2pn$	$2N$	mp^n	mp^nN
$W2$, n.lin. (5.20)	conv./n. conv.	convex	mn	$m + 2N$	–	–

6.1 Properties of the underlying optimization problems

It is worth considering the underlying optimization problems in detail to understand the complexity of the problems to be solved. The complexity depends on the class of the underlying optimization problem, the number of optimization parameters, the number of optimization constraints and the number of derivatives that must be computed if gradient-based optimization algorithms are applied. A particular class of optimization is convex optimization. Convex optimization problems are characterized by the fact that both the objective function and the inequality optimization constraints are convex, whereas the equality optimization constraints are affine. Effective computational methods exist to solve this class of optimization problems numerically, and the search for the global optimum is enhanced, because any local optimum is also globally optimal.

Table 6.1 shows the **properties of the optimization problems** (4.12), (4.24), (5.12) and (5.20). With the exception of the approach of *tracking the vertexes of a polytope* ($W1$) and of *decomposing Solution Space constraints* ($W2$) in conjunction with non-convex Solution Space constraint functions, all objective functions are convex. In the presence of convex inequality optimization constraint functions and affine equality optimization constraints, the optimization problem is convex.

The **number of optimization parameters**, denoted by N , depends linearly on the number of dimensions d in the case of the *box optimization approach based on vertex tracking* ($V1$). However, in the case of 2d-spaces, the number of optimization parameters is large if both the numbers of dimensions and the number of vertexes of the two-dimensional

polygons (approach *W1*) and the number of dimensions and the number of Solution Space constraints (approach *W2*) is large respectively. In Table 6.1, d is assumed to be even, and hence the number of 2d-spaces is $n = d/2$.

For all problems, the **number of linear optimization constraints** P scales with the number of optimization parameters N , as each optimization parameter is bounded by a maximum and a minimum value, i.e. $2N$ linear inequality constraints must be considered. In the linear case for approach *V1*, each of the m Solution Space constraints is assigned to a particular vertex of the box, and hence m linear inequality constraints must be considered. To ensure boxes with positive volumes only, d additional inequality constraints must be considered; see constraint (4.14). For approach *W1*, each constraint is assigned to all vertexes of the polytope, and hence, in the linear case, mp^n number of linear inequalities must be evaluated, with p as the number of vertexes of each two-dimensional polygon. Independent of whether linear or nonlinear Solution Space constraints are present, m linear equality constraints must be considered for approach *W2*; see Equation (5.9).

In the nonlinear case, the **number of nonlinear optimization constraints** denoted by Q for approach *V1* varies from m to $m2^d$. For monotone functions, each Solution Space constraint can be assigned to a particular vertex of the box, and hence m constraints must be evaluated. In the case of non-monotone but convex Solution Space constraints, all vertexes must be checked and $m2^d$ nonlinear constraints must be considered. For functions that are non-monotone but convex w.r.t. some but not all design variables, the number of nonlinear constraints lies in between m and $m2^d$. For approach *W1* in presence of nonlinear but convex Solution Space constraints, each Solution Space constraint must be assigned to each vertex of the polytope and hence, mp^n nonlinear optimization constraints must be considered.

The derivative of each of the nonlinear optimization constraints w.r.t. each of the optimization parameters must be computed, either analytically or by e.g. finite-differences. Therefore, the **number of derivatives of the nonlinear optimization constraints** denoted by D for approach *V1* is between $m2d$ and $m2^d2d$ and is mp^n2pn for approach *W1*.

Figure A.3 in Appendix A.3 depicts values of N and P in dependence on the number of dimensions and the number of Solution Space constraints in the linear case, i.e. if linear Solution Space constraints are present only. Figure A.4 shows values of N , P , Q and D in case of nonlinear Solution Space constraints. In both figures, d is assumed to be even and hence $n = d/2$. Furthermore, for approach *W1* the number of vertexes is assumed to be $p = 4$.

6.2 Numerical effort

In general, optimization algorithms comprise the following main parts: The first part is the **computation of a step**, i.e. an update of the optimization parameters for the next iteration, and the second part is the **evaluation of the optimization constraints**. In the case of a gradient-based optimizer, if necessary, a third part is the **estimation of the derivatives**. Additionally, memory must be allocated, variables must be initialized, etc. Figure 6.1 depicts two examples, showing how the different parts contribute to the overall CPU time. (A) is typical for cases where the Solution Space constraint functions are provided as parametric functions and the derivatives are derived analytically, hence the evaluation of the Solution Space constraints is computationally inexpensive and derivatives are provided by the user and consequently need not be estimated. Case (B) occurs if the Solution Space constraint functions are provided as oracle functions, are time consuming to be evaluated and no derivatives are provided to the optimizer.

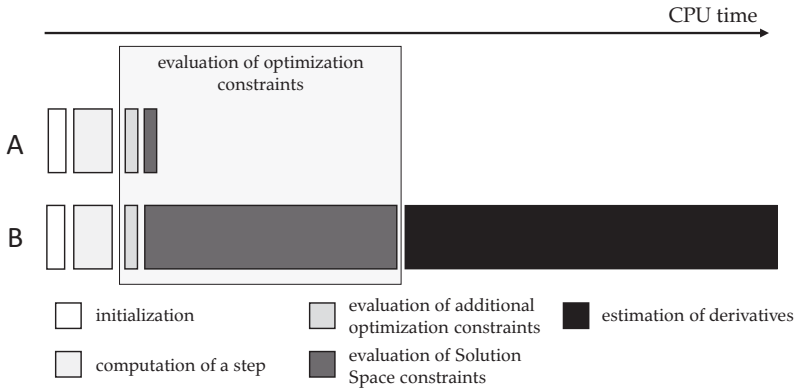


Figure 6.1: Contribution of the different parts of an optimization algorithm to the overall CPU time. In (A), with the exception of the initialization, all parts have the same portion, while in (B) the evaluation of the Solution Space constraints and the estimation of the derivatives are the most time consuming parts.

In the case of using the **interior-point algorithm**, in each iteration the algorithm attempts to compute a Newton step, which requires to solve a linear system, comprising the derivatives of the objective function as well as the derivatives of the optimization constraints; see Appendix A.2. In the interior-point implementation in MATLAB® this is accomplished by applying LDL factorization, with a numerical effort scaling with the size of the linear system. As already mentioned, the latter depends on the number of optimization parameters N and the number of optimization constraints P and Q . Table

6.1 shows how N , P and Q depend on the number of design variables d , the number of Solution Space constraints m , the number of 2d-spaces n and the number of vertexes of two-dimensional polygons p . If a step is rejected, additional evaluations of the objective function and optimization constraints are necessary to adapt the step appropriately. In addition to the computational effort for the determination of a step, the effort for the evaluation of the optimization constraints and the determination of the derivatives, if not provided by the user, must be taken into account.

The **pattern search algorithm** involves no expensive computations to determine a step, see Appendix A.2, but generally requires many evaluations of the objective function and optimization constraints per iteration. In the case of the pattern search implementation in MATLAB[®], a maximum of $2d$ function evaluations per iteration are necessary.

In the case of the **stochastic Solution Space algorithm** for box-shaped Solution Spaces [78], the complexity of the *Trim Algorithm*, which adapts the boundaries such that the box contains good designs only, is $O(\tilde{N}^2d)$ with \tilde{N} as the number of sample points; see Section 4.4.3. For each iteration, each box is evaluated by \tilde{N} sample points. Therefore, the number of overall evaluations of the Solution Space constraints scales with the number of sample points, which is 100 in most of the examinations in [36] and recommended for general problems in [45].

6.2.1 Box-shaped Solution Spaces: Tracking vertexes of a box vs. the stochastic Solution Space algorithm

The overall effort in terms of CPU time for solving the problem of seeking a box with maximum size measure, satisfying the Solution Space constraints, is influenced by the **number of iterations** and the **numerical effort per iteration**. As mentioned above, the numerical effort of an optimization algorithm is mainly composed of the effort for computing a step, the effort for evaluating the optimization constraints, and if a gradient-based optimizer is applied and no derivatives are provided by the user, the estimation of the derivatives. In the following, the approach of using a gradient-based optimizer in conjunction with vertex tracking ($V1$) and the stochastic Solution Space algorithm for box-shaped Solution Spaces ($U1$) are compared.

Number of iterations Table 4.1 and 4.3 show, that in the case of solving problem (4.12) and (4.24) respectively by using the interior-point algorithm and applying the approach of vertex tracking ($V1$), where in the monotone case each Solution Space constraint is assigned to one particular vertex of the box, the *number of iterations* does not depend on the number of dimensions for the linear and monotone test problems ($A1$, $A2$, $B1$, $C1$). All four examples require approximately ten iteration steps. In the case of convex Solu-

tion Space constraints, the number of iterations increases significantly with an increasing number of dimensions. For test problem $D1$, the number of iterations is seven for $d = 2$ and $d = 4$ and 24 for $d = 10$. For $d = 50$, the number of optimization constraints to be considered is too large to be solved practicably.

The results of the examinations for the stochastic Solution Space algorithm ($U1$) in Graff [36] show that the number of necessary iterations increases significantly with the number of dimensions; see Section 4.4.3. Figure 4.7 shows that the Solution Space algorithm requires approximately 25 iterations to solve the ten-dimensional Tilted-Hyperplane problem and approximately 100 iterations for the 50-dimensional Tilted-Hyperplane problem.

Computing a step As mentioned above, when applying a Newton's method based algorithm, as is the case for approach $V1$, the numerical effort for computing a step scales with the number of optimization parameters N and the number of optimization constraints P and Q . Considering Table 6.1, the number of optimization parameters N scales linearly with d and is between four ($d = 2$) and 100 ($d = 50$) for the examples considered in Section 4.5.3 and 4.6.3. In the presence of monotone Solution Space constraints ($A1, A2, B1, C1$), the number of optimization constraints scales linearly with d and m and is between eleven ($d = 2$ and $m = 1$) and 154 ($d = 50$ and $m = 4$). Although the number of optimization parameters ($N = [4, 100]$) and the number of optimization constraints ($P + Q = [11, 154]$), and hence the computational effort for each iteration, differs considerably for the examples considered, the CPU time required to solve the problem differs only slightly. It seems that the CPU time for computing a step is subordinated for the examples considered. In all examples, the number of iterations is on average equal to the number of function evaluations, which indicates that the solver requires no or only a few adaptations of the step. However, in the case of non-monotone but convex Solution Space constraint functions ($D1$), the number of optimization constraints scales with the number of dimensions by 2^d . Considering an example with $d = 50$ and $m = 1$, the number of optimization constraints is $1e15$. Table 4.3 shows that a problem with $d = 4$ is solvable within a practicable CPU time, however, the CPU time increases significantly when the number of dimensions is increased. Additionally, it is worth mentioning, that for the results in Table 4.1 and Table 4.3 the Solution Space constraint functions are provided as parametric functions, computationally inexpensive to evaluate and furthermore, all derivatives are available a priori.

In the case of the stochastic Solution Space algorithm, the step is computed based on \tilde{N} sample points, and the Solution Space constraint functions must be evaluated for each sample point. Subsequently, the Trim Algorithm eliminates bad designs within the box and hence provides the box for the next iteration. The effort is $O(\tilde{N}^2 d)$, however, no information about the CPU time is provided in [36].

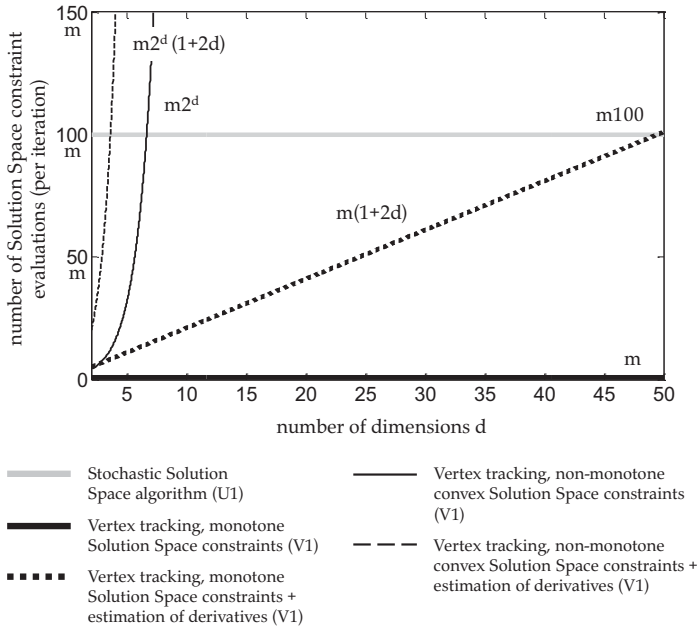


Figure 6.2: Relation between the number of necessary Solution Space constraint evaluations per iteration in dependence on the number of dimensions for the stochastic Solution Space algorithm and the approach using a gradient-based algorithm in conjunction with vertex tracking for monotone and non-monotone but convex Solution Space constraints.

Evaluating optimization constraints Table 6.1 shows that the number of evaluations of the Solution Space constraints per iteration is m in the case of approach *V1* in conjunction with monotone Solution Space constraints and $m2^d$ in conjunction with non-monotone but convex Solution Space constraints. Additionally, d linear inequality constraints, see (4.14), and $4d$ bounds of the optimization parameters must be considered.

In the case of the stochastic Solution Space algorithm, the number of evaluations of the Solution Space constraints per iteration is $m\tilde{N}$, typically $m100$ (with an evaluation of the box by 100 sample points). Note that the information based on the evaluation of the sample points is used for computing a step as well for evaluating the box, e.g. computing \tilde{N}_g/\tilde{N} .

In practice, often the evaluation of the Solution Space constraint functions is the most expensive part, and hence the CPU time for computing a step is subordinated; see Figure 6.1 case (B). If the problem involves monotone Solution Space constraints and the deriva-

tives are provided, approach *V1* requires a hundredth of function evaluations compared to approach *U1*, while the number of necessary function evaluations in presence of non-monotone but convex Solution Space constraints outperform those of approach *U1* in case of $d = 7$; see Figure 6.2 (solid lines).

Estimating derivatives Furthermore, if the gradients of the nonlinear Solution Space constraints are not available in the case of approach *V1*, they must be estimated by e.g. finite-differences. The number of gradients is between $m2d$ in the presence of monotone Solution Space constraints and $m2^d2d$ in the case of non-monotone but convex Solution Space constraints. Hence, the number of function evaluations increases and for each nonlinear optimization constraint, $2d$ further function evaluations are necessary, if forward/backward-finite-differences are applied. This means that approach *V1* in conjunction with monotone Solution Space constraints outperforms approach *U1* in the case of $d < 50$. Figure 6.2 shows the relation between the number of necessary evaluations of m Solution Space constraint functions and the number of dimensions for approach *V1* and *U1*, with and without estimating derivatives. Note that approach *U1* does not require derivatives.

Error of the results Note that for all results obtained by applying approach *V1*, the relative error compared to the analytic solution is negligible.

Figure 4.8 shows that in the case of approach *U1*, the discrepancy between the numerical result and the analytic solution for the Tilted-Hyperplane problem increases significantly with an increasing number of dimensions. However, the fraction of good designs is still close to 100%; see Section 4.4.3.

In summary, if the Solution Space constraints comprise monotone functions or non-monotone convex functions with a moderate number of dimensions, and the derivatives of the Solution Space constraints are available, approach V1 requires a significantly lower number of evaluations of the Solution Space constraints per iteration compared to approach U1. Taking into account that for approach V1 the number of iterations is remarkably lower compared to approach U1, the overall number of evaluations is significantly lower, even if the derivatives of the Solution Space constraints must be estimated by finite-differences. The results obtained by applying V1 show a negligible error compared to the analytic solution, while the error of the results obtained by applying U1 increases significantly with an increasing number of dimensions. This effect is called Vertex Problem, however, in practice, it is advantageous in terms of larger Solution Spaces with a fraction of good designs, which is almost 100%. Furthermore, it is worth mentioning that approach V1 is only applicable to problems with particular types of Solution Space constraints, while approach U1

is applicable to arbitrary problems.

Remark. A comparison between box-shaped Solution Spaces computed by a gradient-based optimizer and by the stochastic Solution Space algorithm, particularly for crash applications, is provided by Fender [28].

6.2.2 2d-spaces: Tracking vertexes of a polytope vs. decomposing Solution Space constraints

For seeking optimal 2d-spaces, optimization problems (5.12) and (5.20) are proposed. The approach of the former is based on the idea of tracking the vertexes of a polytope ($W1$) and the approach of the latter is based on decomposing Solution Space constraints ($W2$).

The result of approach $W1$ is a set of two-dimensional polygons, while the 2d-spaces in case of approach $W2$ may assume arbitrary shape. This implies that only in the cases where the analytic solutions are polygons, both approaches are able to find the identical solution. As an example, consider the Four-Tilted-Hyperplanes problem ($A2$), where each 2d-space assumes the shape of a rectangle. Note that in the case of approach $W1$, the number of vertexes of the two-dimensional polygons must be specified, while the number of vertexes in the case of $W2$ is a result of the optimization. This means that in the case of $W1$, the analytic solution can only be found if the number of vertexes is specified appropriately. For instance, consider the case that the number of vertexes is specified to be three for test problem $A2$. However, the analytic solution are rectangles, and obviously the optimizer cannot provide a result which is identical to the analytic solution. On the other hand, the numerical examinations in Section 5.4 revealed that selecting a higher number of vertexes than the vertexes of the polygon of the analytic solution increases the complexity of the optimization problem remarkably, and often no solution is found.

It is shown that solving problem (5.12) is in general computationally more expensive compared to solving (5.20). Particularly for high-dimensional problems and a large number of vertexes of the two-dimensional polygons, the number of optimization constraints exceeds values of $1e10$; see Figure A.4. This is neither efficiently solvable nor yields accurate results. Furthermore, self-intersecting polygons occur and force the optimizer to recalculate the step, which increases the overall number of function evaluations and often leads to a termination of the optimizer due to a step size that reached a critical value.

Approach $W1$ requires the complete Solution Space to be convex, which is often not the case in real world problems. Approach $W2$ requires the Solution Space constraints to be decomposable, which can be achieved by e.g. approximating the complete Solution Space by parametric models, such as linear models or particular quadratic models. An example is given in Chapter 7.

6.2.3 Box-shaped Solution Spaces vs. 2d-spaces

In general, the effort for the computation of box-shaped Solution Spaces is less compared to the computation of optimal 2d-spaces. This can be seen by comparing the results in Tables 4.1 and 4.3 with the results in Tables 5.1 and 5.2. Due to an increased number of optimization parameters and optimization constraints in the case of approach *W1* and *W2* compared to approach *V1*, see Figure A.3 and A.4, the computational effort increases. Additionally, the underlying optimization problems for seeking 2d-spaces are more challenging to solve, e.g. due to the occurrence of self-intersecting polygons in the case of problem (5.12) (*W1*) and discontinuities in the derivatives of the objective function of problem (5.20) (*W2*). However, 2d-spaces are able to provide significantly more good designs and hence, robustness and flexibility in development processes compared to box-shaped Solution Spaces; see Section 6.3.

6.3 Gain of Solution Space

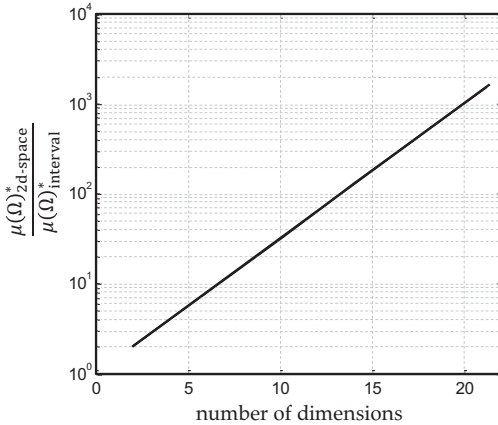


Figure 6.3: Size of an optimal Solution Space based on 2d-spaces divided by the size of an optimal box-shaped Solution Space vs. the number of dimensions, taken from [26].

As shown in Section 3.2.5 and discussed in Section 6.2.3, boxes are, depending on the shape of the complete Solution Space, ill-suited to represent the entire set of good designs. By coupling two design variables pairwise, 2d-spaces are able to provide more good designs. In the following, this is shown by the Tilted-Hyperplane problem introduced

in Section 3.5. Therefore, the introduced size measures (5.3) and (4.2) are considered. They quantify the size of the set of good designs specified by 2d-spaces and intervals respectively. The value of the size measures for the Tilted-Hyperplane problem for an optimal decomposition into 2d-spaces is $\mu(\Omega)_{2d\text{-space}}^* = 0.5^{\frac{d}{2}}$. For an optimal box it can be calculated as $\mu(\Omega)_{\text{interval}}^* = 0.5^d$. Figure 6.3 shows that the ratio of the values of the size measures for optimal 2d-spaces compared to optimal intervals increases significantly with increasing number of dimensions. Consider the case of $d = 20$: The result of the 2d-space approach is able to represent 1,024 times more good designs compared to intervals.

Note that the gain of Solution Space can further be increased by coupling more than two design variables. However, computing 3d-spaces, 4d-spaces, etc. yields results, which are not easily interpretable by decision makers and cannot be visualized in an intuitive manner. Furthermore, in practice, decoupling of requirements is important to accomplish development tasks efficiently; see concurrent engineering [2, 58, 59, 70, 76].

7 — Application

In the following, a chassis design problem, involving typical questions arising in early development stages, is presented. The proposed approaches are applied to demonstrate the applicability and to point out the differences between the approaches. Furthermore, a procedure is presented where the Solution Space is approximated by mathematical surrogates such as linear or quadratic models, Solution Spaces are computed based on the approximation, the results are validated w.r.t. the physical model and, if necessary, the approximation is refined. Additionally, further practicable aspects are considered.

7.1 Chassis design in an early design stage

As mentioned in Chapter 1, the example considered here represents a typical design problem in a stage where conceptual decisions have already been made and chassis components must be designed by simulation to support test engineers in the subsequent development phase; see Figure 1.1. The number of vehicles that must be developed often exceeds 200, differing in the type of the vehicle (sedan, convertible, etc.), the type of the engine (combustion engine, electric engine, etc.), the drive concept (rear-wheel drive, front-wheel drive, all-wheel drive) etc.; see Figure 7.1. From the viewpoint of driving dynamics, the vehicles can be clustered, depending on the desired characteristics of the driving dynamical behavior, i.e. requirements and similarities in the vehicle parameters such as the overall mass of the vehicle and portion of the load acting on the rear axle of the vehicle.

Two examples are presented in the following. **Example One** shows how to apply the methods introduced in this thesis for the development of single vehicles. In **Example Two** a set of vehicles must be designed, e.g. a mid-sized sedan with rear-wheel drive *comprising different combustion engines* (see Figure 1.6), and hence variations in vehicle parameters such as the overall mass and the portion of the load acting on the rear axle of the vehicle must be considered. Functional properties of the main chassis components (anti-roll bars, bump stops, etc.) must be developed, such that the vehicles satisfy specified requirements on stationary and dynamic driving behavior. In *Example One*, all properties of other components that show interactions with the chassis components that must be designed, are assumed to be specified. In *Example Two*, properties of influencing components and subsystems, e.g. the body of the vehicle, are not fully specified and variation must be taken into account.

7.1.1 Design variables and vehicle parameters

Figure 7.2 depicts the front axle of a passenger vehicle and the components that must be designed in the following example. The components are the anti-roll bar, the bump stop and the rebound stop. The anti-roll bar is a torsional spring element, which is connected with the left and right axle and acts in cases where the wheel travel on the one side is different to the wheel travel on the other side. The properties that must be designed are the stiffnesses of the anti-roll bars of the front and rear axles respectively. The bump stop is a spring element situated at the piston rod of the damper, which prevents an abrupt stop of the axle under compression, due to geometrical constraints. Depending on the length, the bump stop is activated when a particular vertical wheel travel is reached, i.e. it has no effect in case of wheel travels beyond a particular value. The properties that must be designed are the stiffnesses and the lengths of the bump stops of the front and

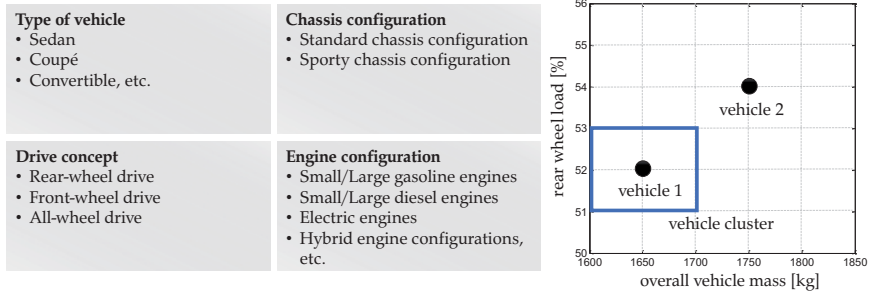


Figure 7.1: In chassis design, typically many vehicles must be considered, differing in the type of the vehicle, the drive concept, engine configuration, etc. This influences, for instance, properties of the overall vehicle such as vehicle mass and rear wheel load. The graph on the right side shows two vehicles differing in the overall mass and rear wheel load (*Example One*) as well as a set of vehicles with a range for each of the two properties (*Example Two*).

rear axle respectively. The counterpart of the bump stop is the rebound stop, its purpose is to prevent abrupt stops of the axle under extension due to geometrical constraints. It is usually located inside the damper tube. The properties that must be designed are the stiffnesses as well as the lengths of the rebound stops of the front axle.

In *Example Two*, the following additional four vehicle parameters, which show variation, are considered: The mass of the overall vehicle $p_{m,veh}$, the portion of the mass, which acts on the rear axle $p_{r1,r}$, the height of the center of gravity above ground $p_{z,cg}$, and the rotatory inertia of the overall vehicle around the vertical axis of the vehicle $p_{i,veh}$. The design variables and vehicle parameters as well as the associated bounds of the design and parameter space are listed in Table 7.1. In *Example Two*, not a single vehicle but a whole set of vehicles must be designed. The vehicles are equipped with different engines. Those with large engines show a higher overall mass, which acts on the front axle and thus the contribution on the rear axle $p_{r1,r}$ is lower. To cover the entire range of different engines, the parameters $p_{m,veh}$ and $p_{r1,r}$ must be varied within the range of [1600, 1700] kg and [51, 53] % respectively. Additionally, based on expected changes in the body of the vehicle, variations of the parameters $p_{z,cg}$ and $p_{i,veh}$ must be taken into account. The ranges are [535, 550] mm and [3050, 3200] kg m² respectively.

7.1.2 Performance measures

In the following examples, the performance of the vehicle regarding its stationary driving behavior as well as regarding its dynamic safety behavior is assessed. Therefore, different standardized maneuvers exist, which allow an objective and reproducible assessment of

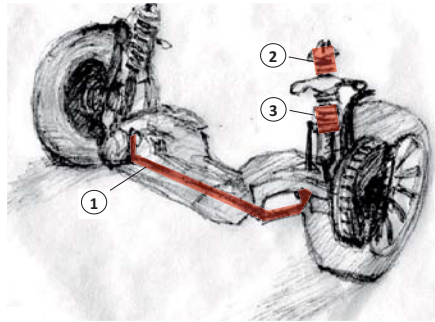


Figure 7.2: Front axle of a passenger vehicle with chassis components anti-roll bar (1), bump stop (2) and rebound stop (3).

the performance of vehicles.

One of the stationery maneuvers is the **quasi-steady-state-cornering** maneuver (QSSC); see [52]. A vehicle follows a circle with a particular radius, e.g. $105m$, while the velocity is slowly increased. With increasing velocity, the lateral acceleration increases to a certain point, where the vehicle can no longer follow the circle. The performance measures considered here, which can be derived from this maneuver, are the ratio of the roll moment acting on the front axle and the total roll moment for a particular lateral acceleration, the roll angle of the vehicle for a particular lateral acceleration and the lateral acceleration at the point where the vehicle can no longer follow the circle (maximum reachable lateral acceleration). The roll moment ratio characterizes the understeering/oversteering tendency of a vehicle. The roll moment is the result of the lateral force acting in the center of gravity while cornering and the associated lever arm between the center of gravity and the height of the roll center. The larger the roll moment ratio, the greater the tendency of the vehicle to show an understeering behavior, which is preferred in chassis design. The roll angle is preferred to be low, due to comfort and safety purposes, while the maximum lateral acceleration is preferred to be high, due to driving dynamical purposes (sporty character).

Here, the **sine-with-dwell** maneuver (SWD) is used to assess the dynamic safety behavior; see [1]. The maneuver mimics a situation where a lane change is executed due to an obstacle ahead. For the SWD maneuver, the steering angle as well as the velocity are specified. The performance measures considered here are the maximum side-slip angle as well as the minimum vertical tire forces. The side-slip angle is the angle between the longitudinal axis of the vehicle and the vector of the velocity of the vehicle. Low values of the side-slip angle and high values of the minimum vertical tire forces are preferred. The reason for the former is the demand on good controllability by the driver, and the reason

Table 7.1: Design variables and vehicle parameters as well as the associated bounds of the design space and parameter space respectively.

design variable	lower bound	upper bound	unit	description
$x_{ca,f}$	15	29	N/mm	Stiffness of the anti-roll bar of the front axle
$x_{ca,r}$	0.5	8	N/mm	Stiffness of the anti-roll bar of the rear axle
$x_{cbs,f}$	10	40	N/mm	Stiffnesses of the bump stops of the front axle
$x_{lbs,f}$	60	110	mm	Lengths of the bump stops of the front axle
$x_{cbs,r}$	10	40	N/mm	Stiffnesses of the bump stops of the rear axle
$x_{lbs,r}$	60	92	mm	Lengths of the bump stops of the rear axle
$x_{crbs,f}$	20	40	N/mm	Stiffnesses of the rebound stops of the front axle
$x_{lrbs,f}$	134	154	mm	Lengths of the rebound stops of the front axle
$p_{m,veh}$	1600	1750	kg	Overall mass of the vehicle
$p_{rl,r}$	50	55	%	Ratio of the mass acting on the rear axle to the overall mass
$p_{z,cg}$	530	570	mm	Height of the center of gravity above ground
$p_{i,veh}$	3000	3300	kg m ²	Inertia of the vehicle around the vertical axis of the vehicle

for the latter is the demand on low roll-over tendency of the vehicle in severe lane change situations.

7.1.3 Physical relations between design variables and performance measures

In the following, the tentative influence of the anti-roll bars, bump stops and the rebound stop on the stationary and dynamical performance measures is briefly explained. All of the design variables in this example have in common that they influence the vertical stiffness of the front and rear axle respectively. As mentioned above, the bump stop and the rebound stop are activated only in the case of vertical wheel travel with a particular magnitude. This implies that the vertical stiffness of the associated axle can not only be increased by increasing the stiffness of the bump stop or the rebound stop, but also by increasing the lengths of the spring elements. The longer the elements the earlier they become active, and hence contribute to the vertical stiffness. The higher the vertical stiffness of the front axle compared to the vertical stiffness of the rear axle, the greater the roll moment ratio and vice versa. The opposite relation holds true for the maximum lateral acceleration while cornering. Increasing the overall vertical stiffness, i.e. the sum of the vertical stiffnesses of the front and of the rear axle, means decreasing the roll angle of the body of the vehicle and vice versa. The performance measures derived from the dynamic maneuver also depend on the vertical stiffnesses. With increasing stiffness of the front axle compared to the rear

Table 7.2: Vehicle performance measures and the associated requirements represented by lower and/or upper bounds.

perf. mea- sure	lower bound	upper bound	unit	description, maneuver
z_R	55	60	%	Ratio of the roll moment acting on the front axle and the total roll moment while cornering with specified lateral acceleration, QSSC.
z_ϕ		2.8	deg	Roll angle of the body while cornering with specified lateral acceleration, QSSC.
z_{ay}	9.20		m/s ²	Maximum reachable lateral acceleration while cornering, QSSC.
z_β	-	12	deg	Maximum side-slip angle while severe lane change maneuver, (here: stability control inactive), SWD.
z_{fz}	800	-	N	Minimum of the vertical tire forces on the left side of the vehicle, SWD.

Table 7.3: Relations between changes in the design variables and changes in the performance measures. To increase the performance measure (+), the value of the associated design variable must be increased (+) or decreased (-).

	$x_{ca,f}$	$x_{ca,r}$	$x_{cbs,f}$	$x_{lbs,f}$	$x_{cbs,r}$	$x_{lbs,r}$	$x_{crbs,f}$	$x_{lrbs,f}$
z_R+	+	-	+	+	-	-	+	+
$z_\phi+$	-	-	-	-	-	-	-	-
$z_{ay}+$	-	+	-	-	+	+	-	-
$z_\beta+$	-	+	-	-	+	+	-	-
$z_{fz}+$	+	-	+	+	-	-	+	+

axle, the side-slip angle will decrease, while the minimum tire forces will increase. The aforementioned is summarized in Table 7.3.

7.1.4 Physical simulation model and mathematical surrogates

For the numerical simulation of the maneuvers mentioned in Section 7.1.2, a look-up table based nonlinear two-track model, implemented in MATLAB[®], is used; see [44]. The model comprises five rigid parts, i.e. the body with six degrees of freedom as well as the tires of the vehicle. For modeling the dynamics of the tires, the Magic Formula model is applied; see [54]. Its name is based on the fact that the equations of the model are not physical but empirical. The coefficients of the model are adjusted such that the forces and moments resulting from the model match best with experimental data. The interaction between the body and the tires is given by look-up tables derived from a high-fidelity multi-

body simulation model in ADAMS®. Once the look-up tables are derived, the model can provide comprehensive results within a few minutes of CPU time for many maneuvers typically considered in chassis design, e.g. quasi-steady-state-cornering and sine-with-dwell. The equations of motion are solved in time domain by numerical integration. The model can be considered as an oracle function, since there is no solution in closed form; see Section 3.2.2.

In the following, mathematical surrogates (also called response surfaces or meta-models; see [67]), particularly parametric models, are used for two reasons. The first reason is that non-gradient-based optimization algorithms require up to hundreds and thousands of function evaluations. Mathematical surrogates are evaluated in milliseconds and hence enable solving the optimization problems within seconds and minutes respectively instead of hours. The second reason is the fact that parametric surrogates, e.g. linear or quadratic models, provide insight into the structure of the system, which enables the analytic computation of derivatives, analytic assessments of the properties of the function and the decomposition of Solution Space constraints, necessary for the approach proposed in Section 5.5.

To generate surrogates, training data must be provided. Thus, the design variables and vehicle parameters listed in Table 7.1 are randomly scattered within the design space and parameter space by Monte Carlo sampling. 10,000 vehicle configurations are generated, simulations with the physical two-track model are executed in parallel (with approx. two days computation time, three computers in parallel) and the performance measures are computed.

On the one hand, a *linear model* is used, which is obtained by applying linear parametric regression to the set of data points. On the other hand, the same data set is used to generate a *quadratic surrogate model*. To apply the approach of decomposing nonlinear Solution Space constraints, a *decomposable quadratic model* with the following particular structure, motivated in Section 5.3, is generated. The coefficients are computed by applying linear parametric regression.

$$g_j(\mathbf{x}) = g_j^1(x_1, x_2) + g_j^2(x_3, x_4) + \dots + g_j^n(x_{2n-1}, x_{2n})$$

with

$$g_j(x_1, x_2) = a_{1,2} + G_{j,1}x_1 + G_{j,2}x_2 + H_{1,1}^jx_1^2 + H_{2,2}^jx_2^2 + H_{1,2}^jx_1x_2$$

$$g_j(x_3, x_4) = a_{3,4} + G_{j,3}x_3 + G_{j,4}x_4 + H_{3,3}^jx_3^2 + H_{4,4}^jx_4^2 + H_{3,4}^jx_3x_4$$

...

The fidelity of the surrogates is assessed by measures well-known from the field of classification; see e.g. [57]. Solution Spaces are related to classification problems based

on the fact that Solution Spaces rely only on the information whether a design is good or bad, i.e. satisfies the requirement on the system or not, and not on the value of the performance function itself. 70% of the data points are considered for the generation of the surrogates, while the remaining data is used for validation. Tables 7.4, 7.5, and 7.6 show the results. Appendix B.1 shows the correlation between the output of the physical model (true output) and the output of the mathematical surrogate for the performance measure z_β . The measures are defined as follows:

- *True positives*: Data points where both the output of the surrogate model and the true output satisfy the associated requirement.
- *True negatives*: Data points where both the output of the surrogate model and the true output do not satisfy the associated requirement.
- *False positives*: Data points where the output of the surrogate model satisfies the associated requirement, while the true output does not satisfy the associated requirement.
- *False negatives*: Data points where the output of the surrogate model does not satisfy the associated requirement, while the true output satisfies the associated requirement.

A model with a high portion of false negatives and a low portion of false positives is said to be conservative w.r.t. the requirements on the system. As a consequence, the Solution Space approximated by those models is smaller than the actual Solution Space, but results obtained by solving the optimization problems proposed in this thesis based on these models do most likely hold true w.r.t. the physical model. Models with a high portion of false positives and a low portion of false negatives, however, yield an approximation of the actual Solution Space, which is larger compared to conservative models, however, results based on these surrogates may fail w.r.t. the physical model.

Figure 7.3 shows a procedure where the actual overall Solution Space is approximated by surrogate models, which enables to compute optimal Solution Spaces very efficiently. Depending on the surrogate, the computed Solution Spaces may contain bad designs and the process must be repeated by generating more conservative surrogates. Ongoing research showed that a support vector machine is able to provide mathematical surrogates of arbitrary order, i.e. linear models, quadratic models, etc., with a specified portion of false positives and false negatives respectively. In the following, the results are validated by generating 100 randomly and uniformly distributed sample points within the Solution Spaces obtained by solving the proposed optimization problems and assessing each point as good or bad w.r.t. the physical model.

As proposed in the sections above, the problem of seeking Solution Spaces can be solved more efficiently in cases, where the Solution Space constraints show particular properties, e.g. are monotone or convex w.r.t. the inputs. To analyze the type of Solution Space constraints in the case of the quadratic model, Equations (3.6) and (3.7) are applied. Therefore, the minimum and maximum values of the first derivatives under consideration of the possible ranges of the design variables $[x_i^{lb}, x_i^{ub}]$ and vehicle parameters are computed by applying interval arithmetic; see Section 4.2.2. Since the first derivatives are linear functions w.r.t. the design variables, the same sign of the minimum value and the maximum value means monotone behavior. An assessment of the quadratic model shows that the performance measures are not monotone w.r.t. some of the design variables. However, this is not in accordance with the explanations based on physical relations in Section 7.1.3, but may be explained by approximation errors of the quadratic model. Analyzing conditions (3.6) and (3.7) statistically, i.e. by generating random designs \mathbf{x} and analyzing one-dimensional sections of the high-dimensional design space allows to obtain a **degree of monotonicity**. The idea is that if a function shows non-monotone behavior for very few combinations of input variables, i.e. designs \mathbf{x} , it is very unlikely that the final box reaches these non-monotone regions of the input space. This allows to apply vertex tracking where each Solution Space constraint is assigned to a particular vertex of the box, even if the functions are not monotone for all designs within the design space. The results of the analysis for *vehicle 1*, where 1,000 designs are generated randomly and uniformly within the design space, show that 4% of the one-dimensional sections of the performance measures z_ϕ and z_{ay} are non-monotone w.r.t. the design variable $x_{\text{cbs.f}}$. The results of the analysis for *vehicle 1* with 100 randomly generated designs is provided in the Appendix B.3. Since the occurrence of non-monotone behavior is maximum 4%, vertex tracking with assignment of the Solution Space constraints to particular vertexes is applied to the quadratic model. The analysis is repeated for *vehicle 2* and for the case in which all twelve input variables are varied within their associated design space; see *Example Two*.

Based on the fact that the approach introduced in Section 4.6.1 for monotone functions can be applied, the problem can be solved with very few necessary function evaluations and CPU time. By computing the eigenvalues of the Hessian of the quadratic model, see Definition 8, it can be shown that the Solution Space constraints are neither convex nor concave. As mentioned in Section 5.4, the approach of vertex tracking of a polytope can only be applied to convex problems, and hence it is not applied to the chassis design problem approximated by the quadratic model.

7.1.5 Technical questions

In the following, the proposed approaches are applied to give answers to the following typical questions, which arise in chassis design in early development stages:

Table 7.4: Percentage of data points that are of the category *true positive*, *true negative*, *false positive* and *false negative* for the **linear model**. The model is generated based on a set of 7,000 data points and validated by 3,000 additional data points.

	true positive	true negative	false positive	false negative
$z_R > 55$	75.4	20.7	1.9	2.0
$z_R < 60$	79.6	17.0	2.3	1.1
$z_\phi < 2.8$	26.9	69.1	1.4	2.6
$z_{ay} > 9.2$	91.1	4.3	2.4	2.2
$z_\beta < 12$	47.9	48.2	1.4	2.5
$z_{fz} > 800$	45.1	50.5	1.3	3.1

Table 7.5: Percentage of data points that are of the category *true positive*, *true negative*, *false positive* and *false negative* for the **quadratic model**. The model is generated based on a set of 7,000 data points and validated by 3,000 additional data points.

	true positive	true negative	false positive	false negative
$z_R > 55$	77.0	22.2	0.4	0.5
$z_R < 60$	80.4	18.8	0.4	0.4
$z_\phi < 2.8$	29.0	69.6	0.9	0.5
$z_{ay} > 9.2$	91.9	4.3	2.4	1.4
$z_\beta < 12$	49.6	49.0	0.5	0.8
$z_{fz} > 800$	47.5	50.4	1.4	0.7

Table 7.6: Percentage of data points that are of the category *true positive*, *true negative*, *false positive* and *false negative* for the **decomposable quadratic model**. The model is generated based on a set of 7,000 data points and validated by 3,000 additional data points.

	true positive	true negative	false positive	false negative
$z_R > 55$	76.1	21.6	0.9	1.4
$z_R < 60$	80.1	18.1	1.1	0.7
$z_\phi < 2.8$	27.7	69.7	0.8	1.8
$z_{ay} > 9.2$	91.4	4.3	2.4	1.9
$z_\beta < 12$	48.7	48.8	0.7	1.7
$z_{fz} > 800$	46.1	50.8	1.0	2.1

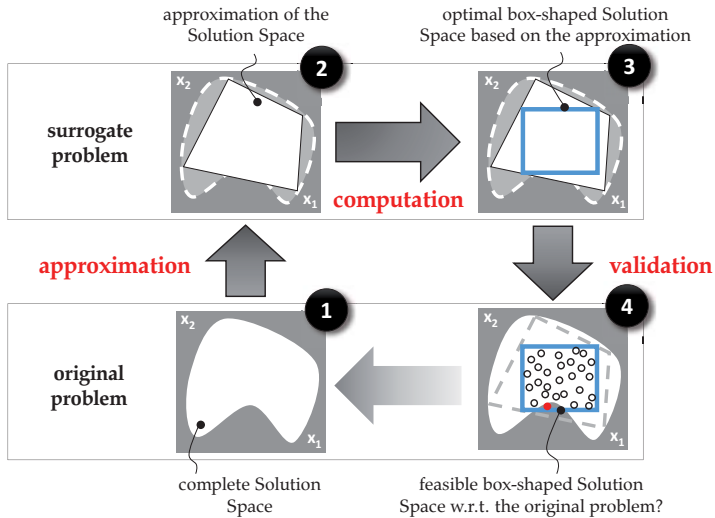


Figure 7.3: Procedure of using surrogates for the computation of Solution Spaces. Surrogates with particular properties, e.g. linear, monotone, convex, etc. enable an efficient computation of Solution Spaces.

1. Which (combinations of) chassis component properties are permissible in the sense that all requirements on the vehicle are satisfied? In *Example Two*, additionally, the following must be considered: The vehicle parameters listed in Table 7.1, e.g. the overall mass of the vehicle, scatter in a specified range. *Solution*: Computing Solution Spaces for the design variables. For *Example Two*, additional constraints must be considered in order to fix the Solution Spaces of the vehicle parameters to the specified ranges. **(Question One)**
2. Is it possible to obtain a larger Solution Space for particular chassis component properties, e.g. the stiffnesses of the anti-roll bars of the front and the rear axle? *Solution*: Computing Solution Spaces and weighting the size measures associated to particular design variables. **(Question Two)**
3. Which ranges of particular chassis component properties, e.g. those of the bump stops, can be excluded for further considerations, since they will not lead - independently of the values of the other properties - to a system that satisfies all requirements? *Solution*: Computation of the outer box. **(Question Three)**

7.2 Numerical results of Example One (single vehicles)

In *Example One*, two single vehicles (see Figure 7.1) must be designed with the design variables listed in Table 7.1, such that the requirements provided in Table 7.2 are satisfied. For the 2d-spaces, design variables must be paired. Generally, this can be done arbitrarily, however, some pairings might lead to larger Solution Spaces compared to others, and hence this might be considered as a further degree of freedom for the underlying optimization problem; see also [26]. In this example, design variables that belong to one component are paired to enable independent design work.

Firstly, the eight design variables of a vehicle with the following vehicle parameters must be designed: $p_{m,veh} = 1650$ kg, $p_{rl,r} = 52$ %, $p_{z,cg} = 540$ mm, $p_{i,veh} = 3125$ kg m² (vehicle 1, Figure 7.1). The proposed approaches are applied, namely the interval approach based on a stochastic Solution Space algorithm (*U1*) (black boxes, thin lines), based on vertex tracking (*V1*) (blue boxes, bold lines), the 2d-space approach based on vertex tracking of a polytope (*W1*) (black polygons, dashed lines) and based on decomposing Solution Space constraints (*W2*) (white areas). The results are based on the *linear model*, the *quadratic model* and, in the case of approach *W2*, on the *decomposable quadratic model*. To show the validity of the results, 100 sample points are randomly and uniformly distributed within the obtained Solution Spaces and the performance measures are computed and evaluated on the physical model.

Secondly, the four design variables of the bump stops of the front and rear axle of a vehicle with the following vehicle parameters must be designed: $p_{m,veh} = 1750$ kg, $p_{rl,r} = 54$ %, $p_{z,cg} = 565$ mm, $p_{i,veh} = 3225$ kg m² (vehicle 2, Figure 7.1). The focus in this four-dimensional example is to give an answer to *Question Three*, stated in Section 7.1.5, by computing the outer box. The design variables of the anti-roll bar of the front and of the rear axle as well as the design variables of the rebound stop of the front axle are specified by: $x_{ca,f} = 24$ N/mm, $x_{ca,r} = 4$ N/mm, $x_{crbs,f} = 28$ N/mm, $x_{lrbs,f} = 140$ mm. Additionally, the optimal box as well as the optimal 2d-spaces are computed. All results are based on the *linear model*.

Answer to Question One Figures 7.4 and 7.5 show the results for the eight-dimensional chassis design problem. Both the linear model and the quadratic model (for approach *W2* the decomposable quadratic model is used) are applied. The results give an answer to *Question One*, stated in Section 7.1.5. In the case of the linear model, the optimal Solution Spaces based on approach *U1*, *V1*, *W1* and *W2* are provided. In the case of the quadratic model, approach *W1* is not applied, because the quadratic model is not convex, and hence does not satisfy the condition for vertex tracking in conjunction with polytopes; see Section 5.4. The portion of sample points that satisfy all requirements

evaluated on the physical model is provided in Table 7.9. Table 7.7 provides the numerical effort in terms of number of iterations, number of function evaluations, CPU time as well as number of optimization parameters and optimization constraints. Note that the results obtained by applying the stochastic Solution Space algorithm are not deterministic. The values provided in Table 7.7 for approach *U1* are the mean values based on five subsequent runs; see Appendix B.4. The initial box was determined by seeking a single feasible design and considering the design as center point of the initial box, with each edge length being 10% of the design space; see Section 4.5.2. The initial box is $\mathbf{x}_0^l = (27.58, 6.91, 31.94, 83.76, 15.18, 85.93, 20.02, 140.39, 1650, 52, 540, 3125)$ and $\mathbf{x}_0^u = (28.98, 7.66, 34.94, 88.76, 18.18, 89.13, 22.02, 142.39, 1650, 52, 540, 3125)$. Table 7.8 shows the numerical effort for the results based on the quadratic model. The procedure to determine the initial box is the same as for the linear model, the initial box is $\mathbf{x}_0^l = (27.56, 7.24, 36.97, 86.49, 19.85, 80.64, 27.01, 144.01, 1650, 52, 540, 3125)$ and $\mathbf{x}_0^u = (28.96, 7.99, 39.97, 91.49, 22.85, 83.84, 29.01, 146.01, 1650, 52, 540, 3125)$.

Table 7.7: Numerical effort for the computation of the results provided in Figure 7.4 based on the **linear model**. The number of iterations for approach *U1* is the sum of the number of iterations in the Exploration Phase and the number of iterations in the Consolidation Phase. The overall number of optimization constraints is the sum of the linear inequality constraints, linear equality constraints and the bounds of the optimization parameters.

	no. iterations	no. fun. eval.	CPU time [sec]	no. opt. parameters	no. opt. constraints
<i>U1</i>	50+35*	8500*	32.34*	-	-
<i>V1</i>	10	11	0.09	16	$(6 + 8) + 0 + 2 \cdot 16$
<i>W1</i>	28	32	2.82	32	$1536 + 0 + 2 \cdot 32$
<i>W2</i>	101	490	37.41	24	$0 + 6 + 2 \cdot 24$

*average based on five runs

Answer to Question Two To answer *Question Two*, stated in Section 7.1.5, a weighting factor of ten is used for the size measure associated with the design variables $x_{ca,f}$ and $x_{ca,r}$. Figure 7.6 and Figure 7.7 depict the results based on the linear model and the quadratic model. For approach *W2*, the decomposable quadratic model is applied. The results of approach *V1*, *W1*, and *W2* are shown. In the case of the quadratic model, approach *W1* is not applied, because the quadratic model is not convex and hence does not satisfy the condition for vertex tracking in conjunction with polytopes; see Section 5.4. Approach *U1* is not able to weight particular size measures and hence is not applied here. As initial box the one used to answer *Question One* is used.

Table 7.8: Numerical effort for the computation of the results provided in Figure 7.5 based on the **quadratic model**. The number of iterations for approach *U1* is the sum of the number of iterations in the Exploration Phase and the number of iterations in the Consolidation Phase. The overall number of optimization constraints is the sum of the nonlinear inequality constraints, nonlinear equality constraints, linear inequality constraints, linear equality constraints and the bounds of the optimization parameters.

	no. iterations	no. fun. eval.	CPU time [sec]	no. opt. parameters	no. opt. constraints
<i>U1</i>	50+35*	8500*	43.60*	-	-
<i>V1</i>	26	29	1.21	16	6 + 0 + 8 + 0 + 2 · 16
<i>W1</i>	-	-	-	-	-
<i>W2</i>	250	8071	292.02	24	0 + 0 + 0 + 6 + 2 · 24

* average based on five runs

Table 7.9: Portion of good sample points in % within the computed Solution Spaces of the different approaches evaluated on the physical model for the eight-dimensional chassis design problem, *Example One*, providing an answer to *Question One*; see Figure 7.4 and Figure 7.5

	<i>linear model</i>	<i>quadratic model</i>
<i>U1</i>	99	98
<i>V1</i>	100	100
<i>W1</i>	100	-
<i>W2</i>	100	100

Answer to Question Three To answer *Question Three*, stated in Section 7.1.5, for the four-dimensional problem, the outer box is computed by applying the linear model. Therefore, $\binom{6+8}{4} = 1001$ linear equations are solved to compute the vertexes of the four-dimensional polytope, as proposed in Section 4.5.1. Additionally, approach *V1*, *W1*, and *W2* are applied to the linear model. Figure 7.8 shows the optimal box (blue box) as well as the optimal 2d-spaces. Additionally, a box near the boundary of the outer box (small red box) is computed, by setting additional constraints, see Section 4.7.

Discussion Table 7.9 shows that the portion of good sample points, evaluated on the physical model, is larger than 98% for all results, which is sufficient for design problems in an early design stage. Obtaining a value less than 100% can be explained by two effects. Firstly, the approximation of the true complete Solution Space by the surrogate model is inaccurate in terms of false positives and secondly, the optimization results contain

Table 7.10: Normalized size measure of the computed Solution Spaces obtained by the different approaches for the eight-dimensional chassis design problem, *Example One*, providing an answer to *Question One*; see Figure 7.4 and Figure 7.5. Note that for better readability the values are multiplied by $1e5$.

	<i>linear model</i>	<i>quadratic model</i>
<i>U1</i>	17.18	48.04
<i>V1</i>	2.60	20.70
<i>W1</i>	143.12	-
<i>W2</i>	295.04	244.55

constraint violations, particularly for approach *U1* often the *Vertex Problem* occurs; see Section 4.4.3.

Considering the results depicted in Figure 7.4 shows that the intervals computed by approach *U1* are larger compared to the intervals computed by approach *V1*, except for the design variables $x_{\text{crbs},f}$ and $x_{\text{lrbs},f}$. The size measure of the Solution Space computed by approach *U1* is larger compared to the size measure of the Solution Space computed by approach *V1*, the values are $\mu(\Omega)_{U1,\text{lin}} = 1.72e-4$ and $\mu(\Omega)_{V1,\text{lin}} = 2.60e-5^1$; see also Table 7.10. This can be explained i.a. by the *Vertex Problem*; see Section 4.4.3. The stochastic Solution Space algorithm terminated due to 100 out of 100 good sample points in the Consolidation Phase; see Appendix B.4. It is worth mentioning that the results obtained by approach *U1* are not deterministic and each run provides dissimilar results; see Figure B.7 in the Appendix.

A comparison of the optimal 2d-spaces depicted in Figure 7.4 obtained by applying approach *W1* and approach *W2* shows that both approaches provide similar results. However, the size measure of the results obtained by approach *W1* is smaller than the size measure obtained by approach *W2*. This is based on the fact that in the case of *W1*, the number of vertexes of each two-dimensional polygon must be specified, which is $p = 4$ for the example considered. Hence, the optimizer is not able to find the same solution as found by approach *W2*, where the polygons show up to six vertexes.

A comparison between the obtained intervals and optimal 2d-spaces in Figure 7.4 reveals that 2d-spaces are able to show significantly more good designs than intervals. In the case of the 2d-spaces, the requirements on the design variables are coupled pairwise. However, in the example considered, each pair of design variables is associated with a particular chassis component (except for the anti-roll bars) that can be developed in detail independently. Thus, providing 2d-spaces rather than intervals is not contradictory to concurrent engineering for the example considered.

¹The size measures are normalized w.r.t. the size measure of the design space.

In the following, the size measures of the results based on the linear model, see Figure 7.4, are compared with the size measures of the results based on the quadratic model depicted in Figure 7.5. The comparison of the results of the interval approaches $U1$ and $V1$ shows that the size measures based on the quadratic model are larger compared to the size measures based on the linear model. The normalized size measures are $\mu(\Omega)_{U1,quad} = 4.80e-4$ and $\mu(\Omega)_{V1,quad} = 2.07e-4$; see also Table 7.10. This is in accordance with the understanding that the quadratic model is able to represent the true complete Solution Space better compared to the linear model. Better means that the loss of Solution Space caused by the approximation is lower. The aforementioned only is valid, if the models show a similar portion of false positives and false negatives, i.e. have similar approximation qualities. Both the linear model and the quadratic model show a low portion of false positives, and hence are conservative models, see Tables 7.4, 7.5, and 7.6. However, considering the size measures of approach $W2$, based on the linear model and based on the quadratic model, reveals that the latter is smaller, the values are $\mu(\Omega)_{W2,lin} = 2.95e-3$ and $\mu(\Omega)_{W2,quad} = 2.44e-3$. This indicates, that a better approximation of the complete Solution Space does not necessarily lead to larger boxes and 2d-spaces respectively.

Considering the numerical effort of approach $U1$ and $V1$, listed in Table 7.7 and Table 7.8, shows that the number of iterations, the number of function evaluations and the CPU time of approach $U1$ is significantly higher compared to approach $V1$. This can be explained by the fact that approach $V1$ uses the gradient information of the objective function and optimization constraints provided by the user, enabling the algorithm to converge very quickly to the solution of the problem. Furthermore, the examples considered make advantage of the approach of vertex tracking, with an assignment of each Solution Space constraint to a particular vertex of the box. In cases where the derivatives are not provided by the user or vertex tracking with consideration of particular vertexes only cannot be applied, the numerical effort for approach $V1$ raises significantly or the approach may even fail. Note that the stochastic Solution Space algorithm for box-shaped Solution Spaces can handle arbitrary problems and hence is applicable to a wider range of problems.

A comparison of the results in Table 7.7 and Table 7.8, confirms that the numerical effort for the computation of optimal 2d-spaces by applying approach $W1$ and $W2$ is generally greater compared to the computation of boxes by applying approach $V1$. Approach $W1$ is applied to the linear problem only, because the problem based on the quadratic model is non-convex and the condition for the application of $W1$, i.e. convex Solution Spaces, is not satisfied. Furthermore, it is worth mentioning that approach $W1$ requires a very large number of optimization constraints, whereas approach $W2$ in conjunction with the quadratic model requires a very large number of function evaluations, i.e. evaluations

of the objective function and optimization constraints respectively. The former can be explained by the number of vertexes of the eight-dimensional polytope, which is 2^8 , and the latter by the fact that a non-gradient-based optimizer is applied, namely the pattern search algorithm.

Comparing the results depicted in Figure 7.4 and Figure 7.5 with the results depicted in Figure 7.6 and Figure 7.7 shows the effect of weighting the size measures of the design variables associated with the anti-roll bar of the front and rear axle respectively. This yields more space for variation for the design variables $x_{ca,f}$ and $x_{ca,r}$, however, the Solution Spaces for the other variables decreases significantly.

In Figure 7.8, the result of the computation of the outer box is shown. Gray areas are combinations of design variables, independent of the choice of the values of the other design variables, leading to a vehicle that fails w.r.t. the specified requirements. This is important information for test engineers in subsequent design stages, since components with those properties can be excluded from further considerations. The optimal intervals and 2d-spaces depicted in Figure 7.8 are subsets of the complete Solution Space. This implies that components with design variable values outside these Solution Spaces - but inside the outer box - can also be realized, whereas the vehicle satisfies all requirements. This is illustrated by setting additional constraints to the box optimization problem in terms of fixing the lower boundary of the design variable $x_{lbs,f}$ to a value close to the edges of the outer box. The result (red small boxes) is a box that is significantly smaller compared to the box obtained by solving the problem without additional constraints. This confirms that values outside the optimal Solution Spaces, represented by intervals or 2d-spaces, but inside the outer box, also lead to good designs, however, the size of the Solution Space decreases.

7.3 Numerical results of Example Two (set of vehicles)

Example Two differs from *Example One* in that an entire set (cluster) of vehicles comprising different engines is designed, rather than a single vehicle. Therefore, the vehicle parameters $p_{m,veh}$ and $p_{rl,r}$ must be varied within the range of [1600, 1700] kg and [51, 53] % respectively, and hence the influence affected by different engines, i.e. variation in the overall mass and mass distribution in longitudinal direction, is taken into account. Based on expected changes in the body of the vehicle, variations of the parameters $p_{z,cg}$ and $p_{i,veh}$ must also be considered. The ranges are [535, 550] mm and [3050, 3200] kg m² respectively. In all optimization runs, additional equality constraints are set in order to fix the Solution Spaces of the associated vehicle parameters. Therefore, it is guaranteed that the obtained Solution Spaces of the design variables hold true for the specified set of vehicle parameters, mentioned above. For the 2d-spaces, design variables must be paired. In *Example One*

design variables, which belong to one component, are paired to enable independent design work.

To answer *Question One*, intervals and 2d-spaces are computed by applying the interval approach based on a stochastic Solution Space algorithm (*U1*) (black boxes, thin lines), based on vertex tracking (*V1*) (blue boxes, bold lines) as well as the 2d-space approach based on decomposing Solution Space constraints (*W2*) (white areas). The optimization problem of approach *W1* cannot be solved efficiently in the case of *Example Two*, due to many dimensions and consequently a high number of necessary function evaluations. For the computation of the optimal Solution Spaces, the *linear model* is applied. The results are depicted in Figure 7.9.

Discussion Figure 7.9 confirms that the size measure of the results obtained by approach *U1* is larger compared to the size measure obtained by applying approach *V1*.

Note that the Solution Spaces for the eight design variables is valid for an entire set of vehicles, with vehicle parameter values for $p_{m,veh}$ and $p_{r,l,r}$ within the specified ranges. In addition variation in further properties of the vehicle, here $p_{z,cg}$ and $p_{i,veh}$, affected by expected changes in the properties of the body of the vehicle is taken into account. This enables to develop a large set of vehicles, comprising many design variables and requirements robustly and efficiently.

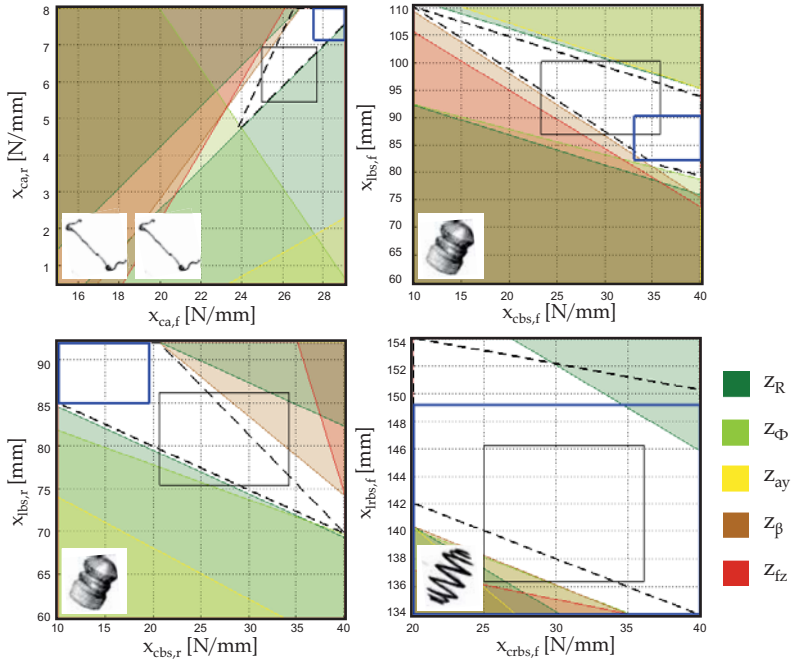


Figure 7.4: The result of the interval approach based on a stochastic Solution Space algorithm (U1) (black boxes, thin lines), based on vertex tracking (V1) (blue boxes, bold lines), the result of the 2d-space approach based on the method of vertex tracking of a polytope (W1) (black polygons, dashed lines) and based on the method of decomposing Solution Space constraints (W2) (white areas, the shaded areas indicate bad regions) for the chassis design problem with eight design variables and six constraints on five vehicle performance measures (Example One, vehicle 1). The results are based on the **linear model** and provide an answer to *Question One*. The vehicle parameters are specified and assume constant values.

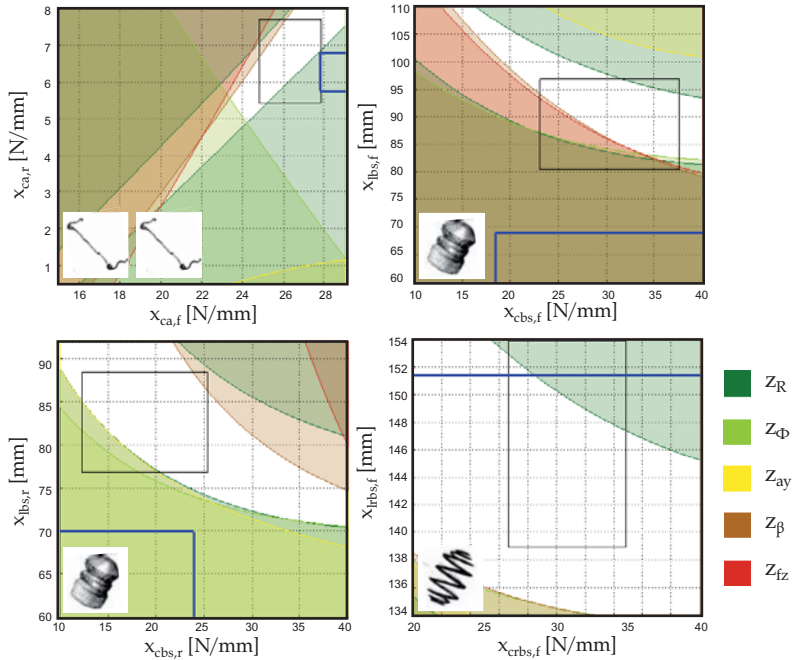


Figure 7.5: The result of the interval approach based on a stochastic Solution Space algorithm (*UI*) (black boxes, thin lines), based on vertex tracking (*VI*) (blue boxes, bold lines) and the results of the 2d-space approach based on the method of decomposing Solution Space constraints (*W2*) (white areas, the shaded areas indicate bad regions) for the chassis design problem with eight design variables and six constraints on five vehicle performance measures (*Example One*, vehicle 1). The results are based on a **quadratic model** and provide an answer to *Question One*. The vehicle parameters are specified and assume constant values.

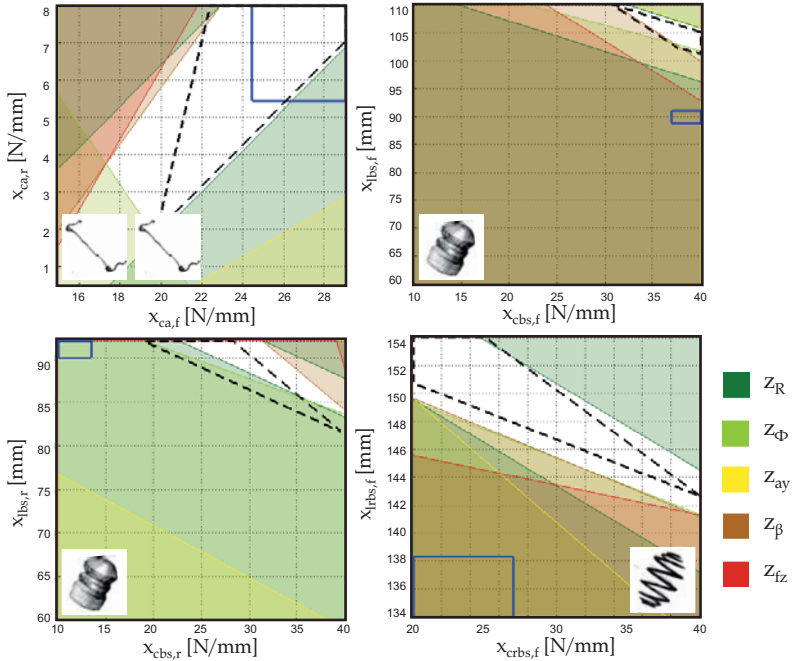


Figure 7.6: The result of the interval approach based on vertex tracking (*V1*) (blue boxes, bold lines), the result of the 2d-space approach based on the method of vertex tracking of a polytope (*W1*) (black polygons, dashed lines) and based on the method of decomposing Solution Space constraints (*W2*) (white areas, the shaded areas indicate bad regions) for the chassis design problem with eight design variables and six constraints on five vehicle performance measures (*Example One*, vehicle 1). The results are based on the **linear model** and provide an answer to *Question Two*. The Solution Space of the design variables of the anti-roll bars are weighted by a factor of 10. The vehicle parameters are specified and assume constant values.

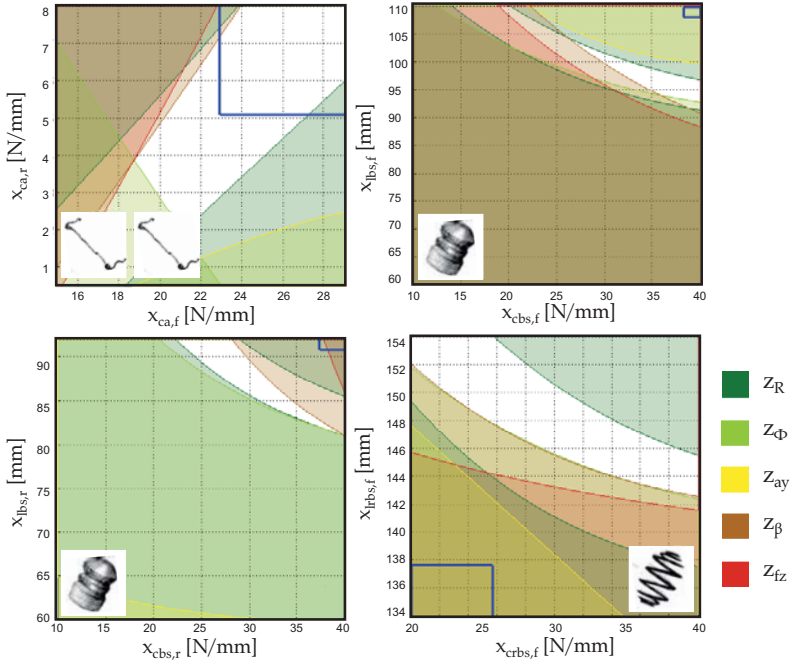


Figure 7.7: The result of the interval approach based on vertex tracking ($V1$) (blue boxes, bold lines) and the result of the 2d-space approach based on the method of decomposing Solution Space constraints ($W2$) (white areas, the shaded areas indicate bad regions) for the chassis design problem with eight design variables and six constraints on five vehicle performance measures (*Example One*, vehicle 1). The results are based on a **quadratic model** and provide an answer to *Question Two*. The Solution Space of the design variables of the anti-roll bars are weighted by a factor of 10. The vehicle parameters are specified and assume constant values.

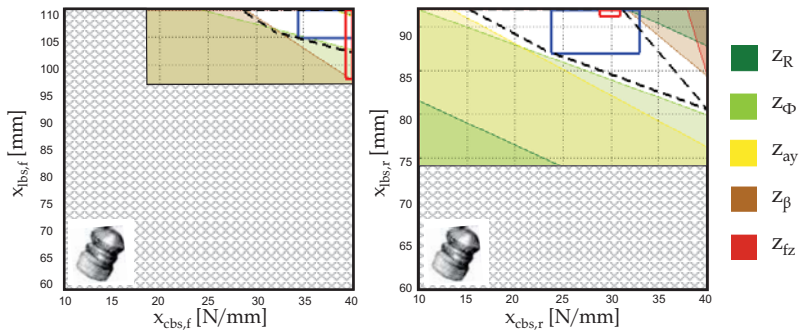


Figure 7.8: The result of the computation of the outer box (gray areas are ranges outside the outer box) as well as the result of the interval approach based on vertex tracking ($V1$) (optimal box: blue rectangles; sub-optimal box: small red rectangles) and the result of the 2d-space approach based on the method of vertex tracking of a polytope ($W1$) (black polygons, dashed lines) and decomposing Solution Space constraints ($W2$) (white areas, the shaded areas indicate bad regions) for the chassis design problem with four design variables and six constraints on five vehicle performance measures (*Example One*, vehicle 2). The results are based on a **linear model** and provide an answer to *Question Three*. All other design variables and vehicle parameters are specified and assume constant values.

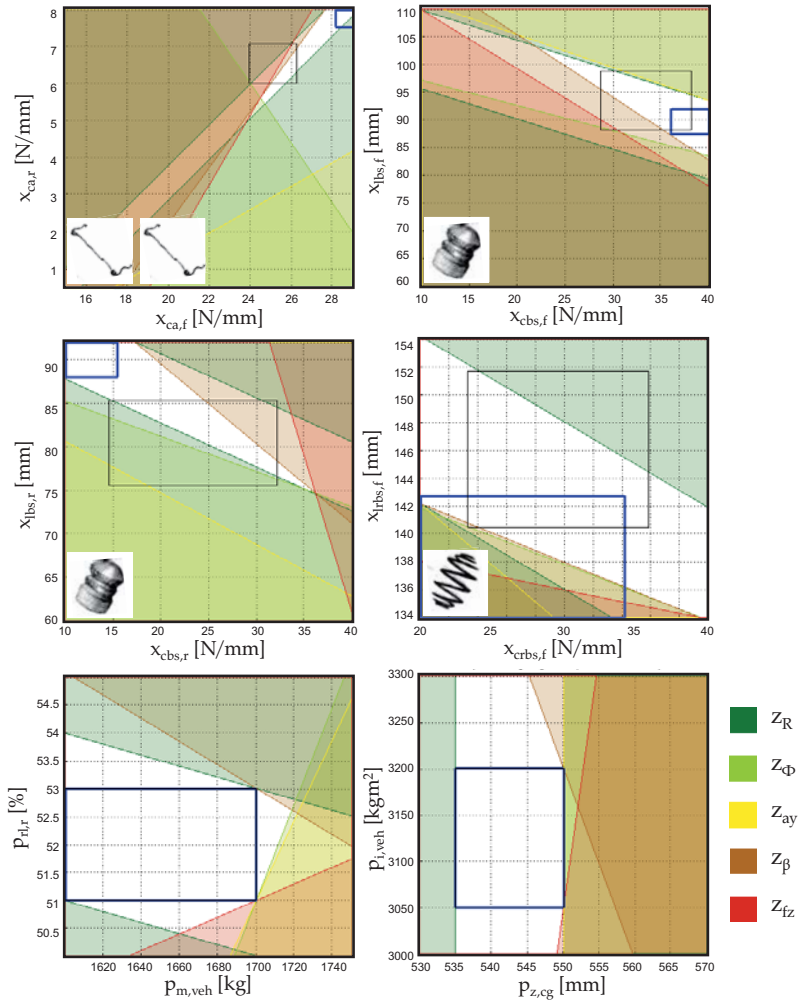


Figure 7.9: The result of the interval approach based on a sampling algorithm ($U1$) (black boxes, thin lines) based on vertex tracking (VI) (blue boxes, bold lines) and the result of the 2d-space approach based on the method of decomposing Solution Space constraints ($W2$) (white areas, the shaded areas indicate bad regions) for the chassis design problem with eight design variables, four vehicle parameters and six constraints on five vehicle performance measures (*Example Two*, set of vehicles). The results are based on a **linear model** and provide an answer to *Question One*. The Solution Spaces of the vehicle parameters are specified to a particular range in order to consider variation due to product family design and lack of information.

8 — Critical reflection

In the first chapters, the necessity for providing sets of designs rather than single designs in early development stages of complex products was motivated, a literature review was conducted and aims and research questions were derived. The following chapter refers back to the aims and research questions stated and provides a critical reflection of what has been achieved.

Aim1: Providing alternative methods for efficient computation of box-shaped Solution Spaces Based on a literature review, the research question reads: *How can box-shaped Solution Spaces with application in chassis design be computed more efficiently compared to existing approaches in terms of computational cost?* In practice, an important measure for computational cost is the overall CPU time for solving numerical problems. The approach to compute box-shaped Solution Spaces by means of vertex tracking (*V1*) in conjunction with a gradient-based optimizer presented in this thesis is benchmarked against a stochastic Solution Space algorithm (*U1*), published by Zimmermann et al. [78]. As long as Solution Space constraints are monotone functions or convex functions in the presence of a low number of dimensions, and derivatives can be provided, approach *V1* is able to solve the problem of seeking box-shaped Solution Spaces with maximum box size measure more efficiently in terms of CPU time compared to approach *U1*. This is particularly the case if the evaluation of the Solution Space constraints is computationally expensive. In industry, however, performance functions and hence Solution Space constraints are often given as black-box functions and neither the properties such as monotonicity and convexity nor derivatives are available. On the basis of an industrial example in Chapter 7, it is shown how mathematical surrogates can be used to approximate the true Solution Space to achieve particular properties and to make derivatives available. For applications with highly nonlinear nature, however, the box cannot be evaluated by assessing the vertexes of the box as it is done in approach *V1*. In this case either approach *V1* must be modified (see outlook) or approach *U1* is applied.

Aim2: Developing alternatives which are able to provide more good designs compared to box-shaped Solution Spaces in order to increase the flexibility and robustness w.r.t. lack of knowledge in development processes Based on a literature review, the research question reads: *How can the set of permissible designs be represented such that the loss of Solution Space is minimum compared to box-shaped Solution Spaces, whereas components can still be developed independently?* In literature, box-shaped Solution Spaces are particularly developed for independent design work in the framework of the V-model approach. In crash application, see [28, 30, 36, 37], requirements on a system level (maximum deceleration during crash) are broken down to a component level (force-deformation characteristic of particular components), which serve as target regions for the detailed design of components (via CAD models and FEM simulations). Box-shaped Solution Spaces and the associated intervals are an intuitive description of design alternatives and decouple requirements on design variables completely. However, this comes with a price in terms of a loss of Solution Space. This means that the box is not able to enclose all good designs, which is depicted via a two-dimensional problem in Figure 4.6. In Figure 6.3 it is shown, that if design variables are coupled (here pairwise) the loss of Solution Space is reduced.

This implies that with a higher degree of coupling, i.e. coupling of two, three, four, five, etc. design variables, the loss of Solution Space tentatively decreases. However, a four-dimensional, five-dimensional, etc. Solution Space (4d-space, 5d-space, etc.) cannot be visualized intuitively. Therefore 2d-spaces are a good trade-off: A gain of Solution Space, whereas an intuitive description of permissible designs is still possible.

In driving dynamics, often one particular component is characterized by two design variables, e.g. the bump stop is characterized by its length and its stiffness and hence the coupling of two design variables still enables independent design work. In Chapter 7, the introduced approaches for the computation of 2d-spaces are applied to an industrial example in chassis design. Six requirements of the system are formulated as requirements on the components, while three components (bump stop of the front and rear axle and the rebound stop of the front axle) can be further developed in detail independently. However, it is also shown, that the computation of 2d-spaces by applying the proposed approaches is in general computationally more expensive than computing box-shaped Solution Spaces. In the case of non-convex Solution Spaces, approach *W1* cannot be applied. In the case of a high number of dimensions and a high number of Solution Space constraints, approaches *W1* and *W2* tend to fail due to a high computational complexity of the underlying optimization problems.

Aim3: Adapting the approaches such that apriori knowledge can be considered in order to make the method of Solution Spaces applicable to a wider range of typical development questions in the field of chassis design

Based on a literature review, the research question reads: *How can the approach of Solution Spaces be extended to incorporate apriori knowledge and hence make it applicable to further questions arising in development?* Solution Spaces are developed to provide space for variation, when components or subsystems are developed in detail. Even in very early development stages, apriori knowledge is often available. Examples are the knowledge that some components show more variation than others, the knowledge that particular components show a specified variation e.g. due to product family design or the knowledge that some components are better in terms of cost, manufacturability, availability, etc. Therefore, the proposed approaches *V1*, *W1* and *W2* are extended to allow taking this into account. Weighting factors allow to yield larger target regions for particular design variables. Additional optimization constraints allow that particular designs are enclosed, which are known to be best in terms of cost, etc. It is worth mentioning that generally using weighting factors or setting additional constraints decreases the overall size measure of the Solution Space, which is depicted in Figure 4.13.

9 — Conclusion

In this chapter, the approaches introduced in this thesis, the results as well as the key findings, are summarized. Furthermore, the main conclusions are pointed out and an outlook on future research work to overcome the limitations mentioned in the previous chapter is given.

Summary and conclusion Many approaches in the field of set-based design exist in literature, and the benefits in various fields of engineering are pointed out in numerous publications. This thesis is concerned with numerical methods for the computation of Solution Spaces, i.e. sets of permissible designs, particularly with application in chassis design.

In the beginning of this thesis, the challenges of designing technical products are explained with focus on chassis design in its early development stages. Challenges arise due to an increased complexity as a result of many components and subsystems interacting with each other, many requirements on the system often in conflict to each other and a large number of related products that need to be designed. Furthermore, in order to provide competitive products, they must be developed efficiently in terms of time and cost. Concurrent engineering is a common used methodology, where components and subsystems are developed in parallel rather than subsequently. However, designing components of complex products simultaneously inevitably increases lack of knowledge, another issue in the development of complex products. Set-based design is able to take uncertainties, particularly due to lack of knowledge, into account. Recently, methods based on numerical simulation were proposed, to compute sets of design alternatives, and hence enabling both efficient and robust design processes.

Therefore, box-shaped Solution Spaces with maximum volume are computed, where each edge of the box represents an interval for each of the design variables, which are properties of components and subsystems. As a result, requirements on the design variables are decoupled from each other in order to enable independent design work and provide space for variation, e.g. necessary due to lack of knowledge. In this thesis, a stochastic Solution Space algorithm published in [78], which is able to handle arbitrary problems, is reviewed. In the past, this was demonstrated by industrial problems in various fields of engineering such as vehicle crash, vehicle driving dynamics, etc. In literature, it is shown that the problem of seeking a box with maximum volume can be solved more efficiently if a gradient-based optimizer is used. However, the published approach [28,29] is applicable for front crash problems only.

In this thesis, the approach published in [28,29] is generalized and analyzed in more detail. The approach outperforms the stochastic approach [78] in the case of particular problems, e.g. monotone problems and problems where the gradient information of the underlying problem is available. Examples from chassis design demonstrate that the approach is applicable to common chassis design problems, and it is shown that problems can be solved more efficiently in terms of a lower CPU time compared to the stochastic algorithm. On the other hand, it is pointed out that the stochastic Solution Space algorithm is not limited to particular problems, and that due to the stochastic nature of the algorithm, box-shaped Solution Spaces are in most cases larger, and hence provide more

space for variation.

A two-dimensional example from chassis design is used to demonstrate that often box-shaped Solution Spaces are ill-suited to represent the entire set of permissible designs. Therefore, two approaches, which provide Solution Spaces for pairwise coupled design variables and as a consequence are able to represent a larger set of permissible designs, are introduced. Both approaches decompose a high-dimensional Solution Space into a set of two-dimensional Solution Spaces. The first approach evaluates the vertexes of a high-dimensional polytope, in order to guarantee that the polytope contains permissible designs only. The second approach decomposes high-dimensional Solution Space constraints into two-dimensional constraints. Numerical examinations confirm that the former approach is in general computationally more expensive compared to the latter approach.

An industrial example comprising eight design variables and four vehicle parameters as well as six requirements on five performance measures demonstrates the benefits of intervals and 2d-spaces respectively. Solution Spaces computed by means of numerical simulation are able to make development processes more efficient in terms of time and cost. For instance, test engineers in subsequent development stages are provided with a set of designs that satisfy all requirements under consideration, and hence the number of test configurations (hardware testing; see Figure 1.1) to be assessed can be reduced significantly. A set rather than a single design, on the other hand, provides test engineers space for exploration, i.e. to assess designs regarding further requirements, which were not available or assessable in earlier stages. Furthermore, Solution Spaces are able to incorporate the effect of unavoidable scatter in interacting components or subsystems and hence, yield robustness. Intervals or 2d-spaces are subsets of high-dimensional Solution Spaces. This implies that there are more good designs outside. Extensions of the approaches are shown, which enable to include preferences of decision makers, e.g. due to information such as cost, manufacturability, availability, etc. In the industrial example considered, more permissible combinations of design variables associated with the anti-roll bars are required, which is realized by a weighting factor. This provides more space for variation for particular properties of components, if necessary. An outer box, with bad designs outside only, is shown to be a useful source of information in design processes. No permissible combinations of design variables can be found outside, and hence those designs can be excluded for further considerations. The example considered shows that often decoupling of requirements for each design variable by means of intervals is not necessary, because several design variables are associated with a particular component. This implies that design variables can be partially coupled, and nevertheless, components can still be designed in detail independently. Aside from the advantage of larger sets of permissible designs, 2d-spaces are intuitive representations of high-dimensional Solution Spaces. In practice, often more than two design variables are associated with a particular component

or subsystem, and it is obvious to extend the approaches to compute 3d-spaces, 4d-spaces, etc. However, the results cannot be represented graphically if more than three design variables are coupled and hence, provide no intuitive representation of design alternatives for decision makers.

In short, the main conclusions are:

- Box-shaped Solution Spaces for chassis design can be computed efficiently in terms of low CPU time for particular problems, e.g. monotone problems, if a gradient-based optimizer in conjunction with vertex tracking (approach *V1*) is used.
- The reviewed stochastic Solution Space algorithm (approach *U1*) for box-shaped Solution Spaces is able to handle a wider range of problems and in most cases provides larger Solution Spaces compared to those where the approach of vertex tracking was applied (approach *V1*). However, generally the overall number of function evaluations is high.
- For the decomposition of high-dimensional Solution Spaces into 2d-spaces two different approaches (approaches *W1* and *W2*), which can be solved by standard optimization algorithms are presented. It is shown that the approaches differ with regard to their numerical complexity and applicability.
- Intervals and 2d-spaces are intuitive representations of high-dimensional Solution Spaces and provide space for variation due to uncertainties, particularly those caused by lack of knowledge. Therefore, Solution Spaces incorporate robustness in design processes.
- Whenever pairwise coupling of design variables is compatible with the design process, the computation of 2d-spaces enables to represent significantly more good designs compared to intervals, and hence provide more robustness/flexibility.
- Solution Spaces computed by means of numerical simulation are an efficient approach to support decision makers in subsequent design stages. For instance, it enables reducing the number of possible configurations for hardware testing significantly, and hence saves time and cost.
- Solution Spaces provide flexibility for decision makers and further designs can be excluded when more information in later development stages is available.

Outlook In Chapter 9 it was pointed out that approach *V1*, which evaluates a box by assessing (assigned) vertexes of the box, is limited to particular problems. For highly nonlinear problems, assessing the vertexes is not sufficient to ensure that the box contains good designs only. Alternatively, the box can be evaluated by generating sample points

(random designs) inside the box, assessing their performance and counting the number of good and bad designs. A non-gradient based standard optimization algorithm can be used to seek the optimal box. Such an approach has the advantage of being applicable to arbitrary problems. It is similar to the stochastic Solution Space approach in [78] in the sense that the box is evaluated by sampling, but it differs in the sense that a new box is generated by a standard optimization algorithm, e.g. a genetic algorithm, instead of a Trim Algorithm used in [78].

In Chapter 9, the limited applicability to convex problems for approach *W1* and to problems with a moderate number of dimensions and Solution Space constraints for *W1* and *W2*, are mentioned. To overcome this limitation, the proposed approach of evaluating the Solution Space by sampling, whereas a standard optimization algorithm is used to seek the optimal Solution Space, can be applied. As a first trial, approach *W1* was modified in the sense that the Solution Space was evaluated by sampling instead of vertex tracking and a genetic algorithm was used. However, first test runs revealed that the genetic algorithm most likely generates self-intersecting two-dimensional polygons, which are not desired in practice.

This motivates extending the stochastic Solution Space algorithm for box-shaped Solution Spaces [78] for the computation of 2d-spaces for arbitrary problems.

It was mentioned that with a coupling of an increased number of design variables, i.e. computing 2d-spaces, 3d-spaces, 4d-spaces, etc., the gain of Solution Space and hence the possibility to enclose more good designs, is increased. Whereas 4d-spaces, 5d-spaces, etc. cannot be visualized, 3d-spaces might be a good choice for problems in which components are characterized by three design variables and are required to be developed in detail independently. For further industrial applications, the approaches for 2d-spaces can also be generalized in the sense that a user specified coupling scheme is considered. For instance, the first design variable is treated as an interval, the second and fourth design variables are coupled pairwise, the third, fifth and sixth design variables are coupled as 3d-spaces, etc.

Mathematical surrogate models are proposed to be used to approximate the complete Solution Space for two reasons: Firstly, if the surrogate model is parametric, it provides insights into the structure of the model and allows an assessment of the properties of the model, e.g. monotone, convex, etc. The latter is important information for the applicability of the approaches proposed. Secondly, mathematical surrogates are evaluated in milliseconds compared to physical models, for which an evaluation might take several minutes or even hours. Figure 7.3 depicts an approach that approximates the complete Solution Space by mathematical surrogates, computes Solution Spaces based on the surrogates and evaluates the obtained Solution Spaces by generating sample points inside. If it turns out that the obtained Solution Space is infeasible w.r.t. the true Solution Space,

the process must be repeated by generating more conservative approximations, i.e. approximations with a small false positive rate. More research must be done to be able to generate approximations with a user specified false positive and false negative rate. First results revealed that an approach based on support vector machines is promising.

A — Optimization

A.1 Karush-Kuhn-Tucker conditions for constrained optimization problems

The Karush-Kuhn-Tucker conditions (KKT conditions) are a generalization of the approach of Lagrange multipliers involving a differentiable objective function and equality constraints to problems comprising inequality constraints. In general, the KKT conditions are necessary for a solution to be optimal. However, for convex optimization problems the KKT conditions are not only necessary but also sufficient. In some cases the optimal solution can be obtained by solving the set of equations analytically. In all other cases, the equations are solved numerically. Considering an optimization problem with the following form:

$$\begin{aligned}
 & \underset{\boldsymbol{\nu}}{\text{minimize}} && f(\boldsymbol{\nu}) \\
 & \text{s.t.} && h_{\text{ineq},j}(\boldsymbol{\nu}) \leq 0, \quad j = 1, \dots, m_{\text{ineq}} \\
 & \text{s.t.} && h_{\text{eq},j}(\boldsymbol{\nu}) = 0, \quad j = 1, \dots, m_{\text{eq}}.
 \end{aligned} \tag{A.1}$$

Let $f, h_{\text{ineq},j}, h_{\text{eq},j} : \mathbb{R}^d \mapsto \mathbb{R}$ be continuously differentiable functions with primal variables $\boldsymbol{\nu}$ and dual variables $\boldsymbol{\lambda}_{\text{ineq}}$ and $\boldsymbol{\lambda}_{\text{eq}}$ respectively. The KKT conditions read (in accordance with Boyd [13])

$$\nabla f(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{ineq}}} \lambda_{\text{ineq},j} \nabla h_{\text{ineq},j}(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{eq}}} \lambda_{\text{eq},j} \nabla h_{\text{eq},j}(\boldsymbol{\nu}) = \mathbf{0} \tag{A.2}$$

$$\begin{aligned}
 & h_{\text{ineq},j}(\boldsymbol{\nu}) \leq 0 \\
 & j = 1, \dots, m_{\text{ineq}}
 \end{aligned} \tag{A.3}$$

$$\begin{aligned}
 & h_{\text{eq},j}(\boldsymbol{\nu}) = 0 \\
 & j = 1, \dots, m_{\text{eq}}
 \end{aligned} \tag{A.4}$$

$$\begin{aligned} \lambda_{\text{ineq},j} &\geq 0 \\ j &= 1, \dots, m_{\text{ineq}} \end{aligned} \tag{A.5}$$

$$\begin{aligned} \lambda_{\text{ineq},j} h_{\text{ineq},j}(\boldsymbol{\nu}) &= 0 \\ j &= 1, \dots, m_{\text{ineq}}. \end{aligned} \tag{A.6}$$

Equation (A.2) is the derivative of the Lagrange function w.r.t. the primal variables, the Lagrange function reads

$$L(\boldsymbol{\nu}, \boldsymbol{\lambda}_{\text{ineq}}, \boldsymbol{\lambda}_{\text{eq}}) = f(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{ineq}}} \lambda_{\text{ineq},j} h_{\text{ineq},j}(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{eq}}} \lambda_{\text{eq},j} h_{\text{eq},j}(\boldsymbol{\nu}). \tag{A.7}$$

Equation (A.2) is comparable to the necessary condition for unconstrained problems, which is $\nabla f(\boldsymbol{\nu}) = 0$, i.e. the gradient must be zero for a point to be optimal. Equation (A.3) and (A.4) state that the primal variables $\boldsymbol{\nu}$ must satisfy the inequality and equality constraints. They are called primal feasibility conditions. Equation (A.5) in conjunction with (A.2) is called dual feasibility condition and (A.6) complementary condition.

The first-order optimality measure, see [53], used in the framework of MATLAB® solvers, is based on the KKT conditions. It is defined as

$$\begin{aligned} \max\{\nabla f(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{ineq}}} \lambda_{\text{ineq},j} \nabla h_{\text{ineq},j}(\boldsymbol{\nu}) + \sum_{j=1}^{m_{\text{eq}}} \lambda_{\text{eq},j} \nabla h_{\text{eq},j}(\boldsymbol{\nu}), \lambda_{\text{ineq},j} h_{\text{ineq},j}(\boldsymbol{\nu})\} \\ j = 1, \dots, m_{\text{ineq}} \end{aligned} \tag{A.8}$$

Note that for cases where f , $h_{\text{ineq},j}$ and $h_{\text{eq},j}$ are not differentiable, semi-smooth Newton's method and bundle methods exist; see e.g. [40].

A.2 Fundamentals of interior-point algorithms and pattern search

A.2.1 Interior-point

The key idea of interior-point algorithms is to transfer an optimization problem comprising inequality constraints into a sequence of problems with equality constraints only, which can be solved by applying Newton's method. Considering a constrained optimization problem (A.1), the approximate problem reads (in accordance with [13])

$$\begin{aligned}
 & \underset{\boldsymbol{\nu}, \mathbf{s}}{\text{minimize}} && f(\boldsymbol{\nu}) - \mu \sum_{j=1}^{m_{\text{ineq}}} \ln s_j \\
 & \text{s.t.} && h_{\text{ineq},j}(\boldsymbol{\nu}) + s_j = 0, \quad j = 1, \dots, m_{\text{ineq}} \\
 & \text{s.t.} && h_{\text{eq},j}(\boldsymbol{\nu}) = 0, \quad j = 1, \dots, m_{\text{eq}}.
 \end{aligned} \tag{A.9}$$

The term $-\sum_{j=1}^{m_{\text{ineq}}} \ln s_j$ is called a logarithmic barrier, with s_j as the slack variable. The logarithmic barrier with $s > 0$ ensures feasibility of the solution. $\mu > 0$ is the barrier parameter and with $\mu \rightarrow 0$ a solution of the approximate problem tends towards the solution of the original problem. In each iteration step with specified value of μ , the primal ($\boldsymbol{\nu}$) and dual ($\boldsymbol{\lambda}_{\text{ineq}}, \boldsymbol{\lambda}_{\text{eq}}$) variables are updated by solving the modified KKT equations. They are obtained by considering the Lagrange function of problem (A.9), which reads

$$L(\boldsymbol{\nu}, \boldsymbol{\lambda}_{\text{ineq}}, \boldsymbol{\lambda}_{\text{eq}}) = f(\boldsymbol{\nu}) - \mu \sum_{j=1}^{m_{\text{ineq}}} \ln s_j + \sum_{j=1}^{m_{\text{ineq}}} \lambda_{\text{ineq},j} (h_{\text{ineq},j}(\boldsymbol{\nu}) + s_j) + \sum_{j=1}^{m_{\text{eq}}} \lambda_{\text{eq},j} h_{\text{eq},j}(\boldsymbol{\nu}). \tag{A.10}$$

As a necessary condition for optimality, the derivatives of the Lagrange function w.r.t. the primal variables and dual variables are required to be equal to zero. This yields the modified KKT conditions, which are equal to those in Appendix A.1, however, the complementary condition is modified to

$$\begin{aligned}
 & \lambda_{\text{ineq},j} h_{\text{ineq},j}(\boldsymbol{\nu}) = -\mu \\
 & j = 1, \dots, m_{\text{ineq}}.
 \end{aligned} \tag{A.11}$$

The set of equalities comprising Equation (A.2), (A.4) and (A.11) can be solved by applying Newton's method. Newton's method is an iterative approach, which is used to find the zero (or approximation of the zero) of a function f . Therefore, the function is linearized at a point x and the following holds

$$f(x + \Delta x) \approx f(x) + \nabla f(x) \Delta x = 0. \tag{A.12}$$

Consequently, seeking the zero of the derivative of f yields

$$\nabla f(x + \Delta x) \approx \nabla f(x) + \nabla^2 f(x) \Delta x = 0 \quad (\text{A.13})$$

with $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$ as a Newton step; see [13]. More general $\Delta(\cdot) = -(A'')^{-1} A'$, with $\Delta(\cdot)$ being the update of variables, e.g. the primal variables and dual variables in the framework of interior-point optimization. The numerical effort to compute a Newton step scales with the size of the linear system to be solved in each iteration, i.e. with the size of the matrix A' and A'' respectively. In the case of problem (A.9), the linear system comprises Equation (A.2), (A.4) and (A.11), and hence depends on the number of optimization parameters N (A.2), the number of equality constraints (A.4) and the number of inequality constraints (A.11) denoted by P (linear constraints) and Q (nonlinear constraints). Within the interior-point implementation in MATLAB® , the system is solved by LDL factorization. Boyds [13] states that the complexity for solving linear systems by LDL factorization is $1/3(N + P + Q)^3$. The computation of the Newton step provides the search direction for the next iterate, while eventually further function evaluations are necessary if a step is rejected and the step size is adjusted appropriately. Different methods for computing the step size exist, e.g. backtracking line search.

Figure A.1 depicts the contour lines of the merit function, used for backtracking line search, which is $f(\nu) - \sum_{j=1}^{M_{\text{ineq}}} \lambda_{\text{ineq},j} h_{\text{ineq},j} - \mu \sum_{j=1}^{M_{\text{ineq}}} \ln(\lambda_{\text{ineq},j} h_{\text{ineq},j}^2 + \epsilon)$, with ϵ as a constant, for three different iteration steps. The objective function of the original problem as well as the inequality constraints are linear. During optimization, the barrier parameter μ is decreased to zero, and the solution of the approximate problem tends to the solution of the original problem. The solution in each iteration step obtained by solving the linear system by Newton's method is marked by a red dot.

Note that the interior-point implementation in MATLAB®, called by `fmincon()`, attempts to take a Newton step. If this is not possible, it applies the method of conjugate gradients in conjunction with trust region. If the approximate problem is not locally convex near the current iterate, the algorithm does not attempt a Newton step but a conjugate gradient step instead. In any case, an LDL factorization must be accomplished, which makes the determination of a step computationally expensive (see MATLAB® documentation on *Constrained Nonlinear Optimization Algorithms*). For more information about the algorithm see [15, 16, 71].

A.2.2 Pattern search

Pattern search algorithms belong to the class of direct search methods also called derivative free methods, since they do not require any information about the derivatives of the objective function and optimization constraints respectively. In order to find the point

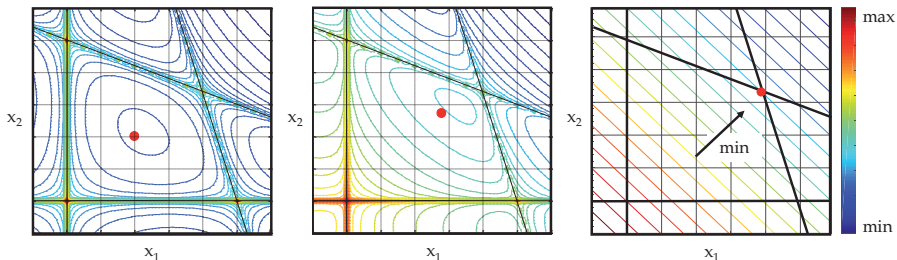


Figure A.1: Applying an interior-point algorithm to a linear problem comprising four linear inequality constraints and a linear objective, which must be minimized. The figures are generated based on an open source MATLAB® implementation of a primal-dual interior-point solver [6] for convex programs with constraints, published by Peter Carbonetto, Department of Computer Science, University of British Columbia; see <https://pcarbo.github.io/convexprog.html>. The barrier parameter μ is decreased to zero throughout the optimization (from left to right, not all iteration steps are shown), and hence the solution of the approximation problem in each step (red dot) converges to the solution of the original problem. The contour lines show the merit function, used for backtracking line search, which is $f(\boldsymbol{\nu}) - \sum_{j=1}^{m_{\text{ineq}}} \lambda_{\text{ineq},j} h_{\text{ineq},j} - \mu \sum_{j=1}^{m_{\text{ineq}}} \ln(\lambda_{\text{ineq},j} h_{\text{ineq},j}^2 + \epsilon)$ with ϵ as a constant.

with best performance, a set of trial points is generated and each point is evaluated. If no constraints are present, the point with the best performance value is chosen and new points are generated. In conjunction with constraints, the feasibility of data points must be taken into account additionally; see e.g. [8]. In the framework of pattern search data points are generated on a mesh, while the mesh size is adapted depending on whether a point with a better performance is found within the actual set or not. The simplest approach to generate data points is to distribute them around one point, two along the direction of each optimization parameter. This is known as coordinate or compass search and generates $2N$ data points, with N as the number of optimization parameters. Many more advanced approaches exist.

Figure A.2 shows a sequence of iteration steps for an unconstrained optimization problem with a quadratic objective function, which is to be minimized. In step 1, data points are generated around an initial guess (red triangle), using the compass search approach. The point with best performance (red square) is selected as the center point for the next step, and the mesh is increased by a factor of 2. If no point with a better performance compared to the center point is found, the mesh size is divided by a factor of 2. The process is repeated until a stopping criterion, e.g. a minimum mesh size, is reached.

By default, the pattern search implementation in MATLAB®, called by the command

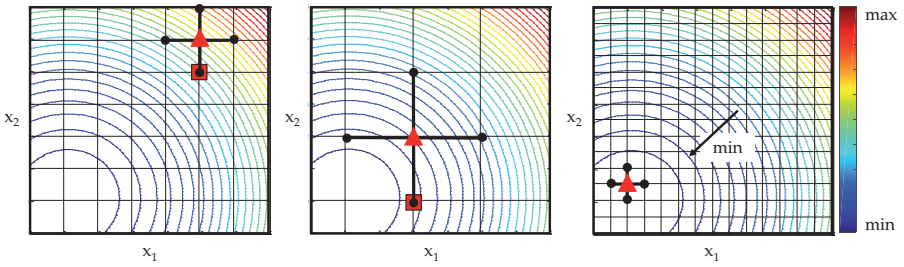


Figure A.2: Different iteration steps (not all iteration steps are shown) of a pattern search algorithm to find the minimum of a sphere function. Black dots mark the trial points around a center point (red triangle). The point with best performance (minimum function value, red rectangle) is taken for the next iteration step as the center point and the mesh size is increased. If none of the points around the center point shows better performance, the center point remains and the mesh size is decreased.

patternsearch(), uses the compass search strategy and generates $2N$ data points per iteration. However, it evaluates each point sequentially and stops if a point is better than the center point. This means that per iteration a maximum of $2N$ data points are evaluated.

Table A.1: The number of optimization parameters N , the number of linear optimization constraints P , the number of nonlinear optimization constraints Q and the number of derivatives of nonlinear optimization constraints D for $d = 10$, $m = 10$, $n = 5$ and $p = 4$.

approach	N	P	Q	D
$V1$, lin. (4.12)	20	60	–	–
$W1$, lin. (5.12)	40	10320	–	–
$W2$, lin. (5.20)	50	110	–	–
$V1$, n.lin. (4.24)	20	50	[10, 10240]	[200, 204800]
$W1$, n.lin. (5.12)	40	80	10240	409600
$W2$, n.lin. (5.20)	50	110	–	–

A.3 Number of optimization parameters, optimization constraints and derivatives

The following figures show the number of optimization parameters N , linear optimization constraints P , nonlinear optimization constraints Q and derivatives D for the approaches $V1$, $W1$, and $W2$ depending on the number of dimensions and the number of Solution Space constraints. Figure A.3 holds true in the case of linear Solution Space constraints. Figure A.4 holds true for nonlinear Solution Space constraints and the number of first-order derivatives, which must be provided to the optimizer, are shown additionally. The values are computed based on the formula presented in Table 6.1. Table A.1 shows the values of N , P , Q and D for $d = 10$, $m = 10$, $n = 5$ and $p = 4$.

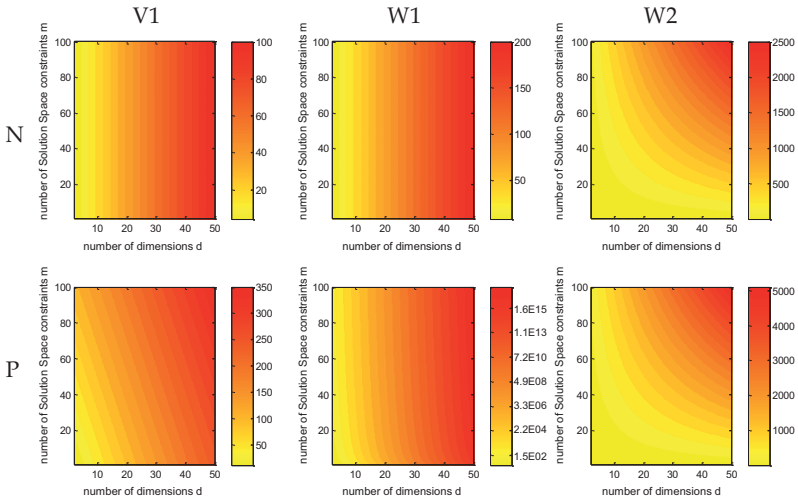


Figure A.3: Visualization of the number of optimization parameters N and linear optimization constraints P for the approaches $V1$, $W1$, and $W2$, assuming d to be even, and hence $n = d/2$. For approach $W1$ the number of vertices is $p = 4$

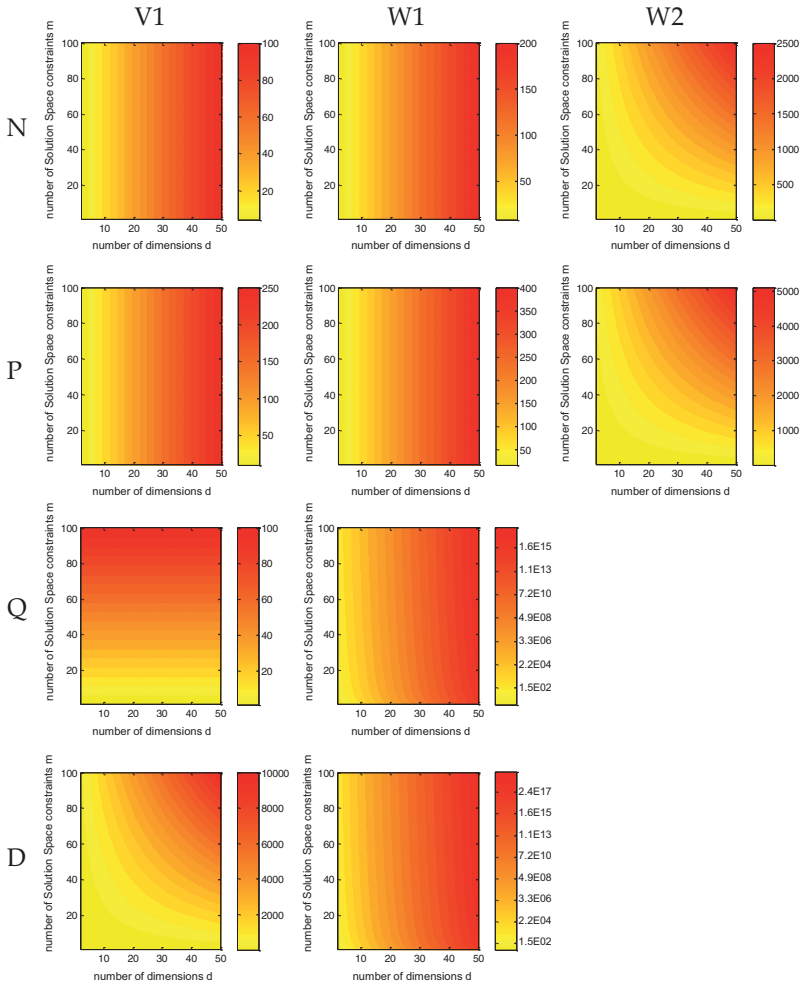


Figure A.4: Visualization of the number of optimization parameters N , linear optimization constraints P , nonlinear optimization constraints Q and derivatives D for the approaches $V1$, $W1$, and $W2$, assuming d to be even, and hence $n = d/2$. For approach $V1$ for Q and D , monotone Solution Space constraints are assumed. For approach $W1$ the number of vertices is $p = 4$.

B — Numerical results

B.1 Classification measures

Figure B.1 depicts the correlation between the true output and the output of the linear and quadratic model for the performance measure z_β . No scatter with all data points on the red diagonal line means that the model's output is equal to the true output, thus the model has a correlation of 1. In the framework of Solution Spaces, however, the quality of the model depends on whether data points, i.e. designs, are classified correctly. For instance, a design that satisfies a requirement (here $z_\beta \leq 12$ deg) is called positive. If the same design satisfies the requirement by considering the output of the approximation model, it is called true positive, and false negative otherwise.

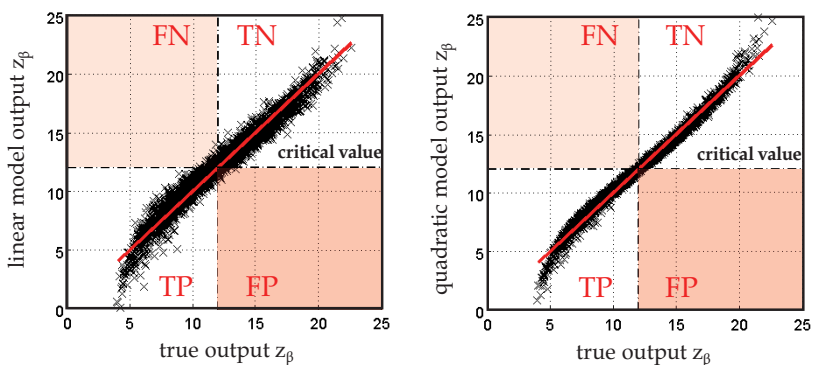


Figure B.1: Correlation between the true output and the output of the linear (left) and quadratic (right) model for the performance measure z_β as well as the quadrants of the classification measures true positive (TP), true negative (TN), false positive (FP) and false negative (FN) for the requirement $z_\beta \leq 12$ deg. The model is generated based on a set of 7,000 data points and validated by 3,000 additional data points.

B.2 Parametric surrogate models

Table B.1: Thresholds and coefficients of the linear model $\mathbf{G}\mathbf{x} \leq \mathbf{g}_c$ (normalized design space $[-1, 1]^d$) for the first six design variables (columns). Note: The coefficients must be multiplied by $1e-2$.

requirement	g_c	$G_{x_{ca},f}$	$G_{x_{ca},r}$	$G_{x_{cbs},f}$	$G_{x_{lbs},f}$	$G_{x_{cbs},r}$	$G_{x_{lbs},r}$
$z_R > 55$	22.7263	-29.4055	28.4701	-7.0377	-21.6033	7.2788	15.3648
$z_R < 60$	26.9598	29.4055	-28.4701	7.0377	21.6033	-7.2788	-15.3648
$z_\phi < 2.8$	-17.3881	-33.6875	-22.3034	-6.1592	-22.5846	-4.4013	-11.6520
$z_{ay} > 9.2$	35.4548	14.8519	-25.1593	5.0400	14.1367	-5.6735	-10.1951
$z_\beta < 12$	-0.0948	-41.2733	29.1829	-19.2625	-28.4274	14.03011	16.3279
$z_{fz} > 800$	-2.2203	-41.3125	23.4168	-16.9612	-26.7313	3.9640	1.1794

Table B.2: Coefficients of the linear model $\mathbf{G}\mathbf{x} \leq \mathbf{g}_c$ (normalized design space $[-1, 1]^d$) for the last two design variables and the four vehicle parameters (columns). Note: The coefficients must be multiplied by $1e-2$.

requirement	$G_{x_{crbs},f}$	$G_{x_{lrbs},f}$	$G_{p_{m,veh}}$	$G_{p_{rl,r}}$	$G_{p_{z,cg}}$	$G_{p_{l,veh}}$
$z_R > 55$	-2,7809	-4,4989	4,1872	14,0920	-0,8383	0
$z_R < 60$	2,7809	4,4989	-4,1872	-14,0920	0,8383	0
$z_\phi < 2.8$	-3.0087	-7.4350	8.9394	-3.4206	16.7937	0
$z_{ay} > 9.2$	-0,4942	-0,5583	11,7091	-5,3877	15,9854	0
$z_\beta < 12$	-2.3768	-5.6118	12.0559	19.2549	2.0872	0.7520
$z_{fz} > 800$	-1.4986	-7.0682	2.1344	-4.7585	14.4105	-1.9742

B.3 Monotonicity

The following figures show how the performance measures change due to a change in one single design variable, which is varied within its associated bounds, i.e. design space. All other design variable values and vehicle parameter values are specified randomly. Hence, the figures show one-dimensional sections of the high-dimensional design space and provide insight into the behavior of the performance measures depending on changes in the design variables. The plots enable statements about the monotonicity of a performance measure w.r.t. particular design variables.

Ratio of the roll moment z_R [%]

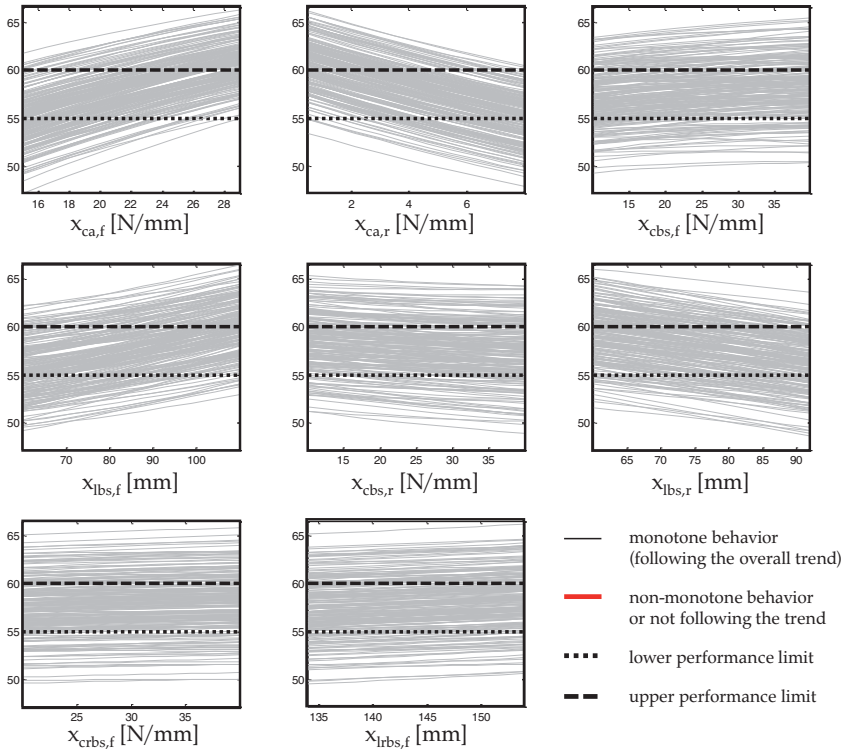


Figure B.2: One-dimensional sections of the twelve-dimensional input space with all other design variables and vehicle parameters assuming constant, randomly chosen values. The performance measure z_R is plotted against each of the eight design variables.

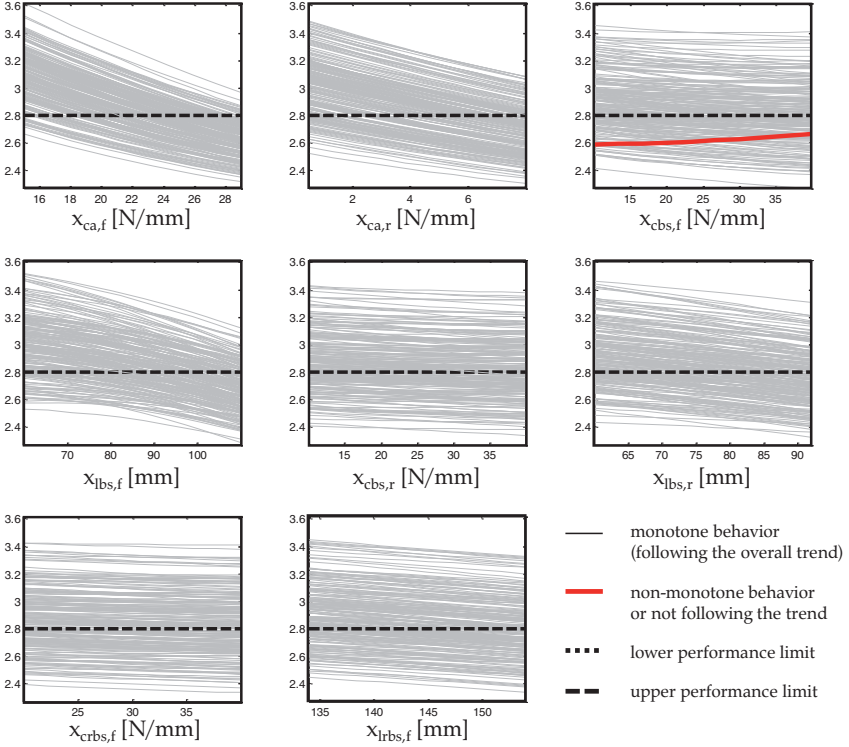
Roll angle of the body z_ϕ [deg]

Figure B.3: One-dimensional sections of the twelve-dimensional input space with all other design variables and vehicle parameters assuming constant, randomly chosen values. The performance measure z_ϕ is plotted against each of the eight design variables.

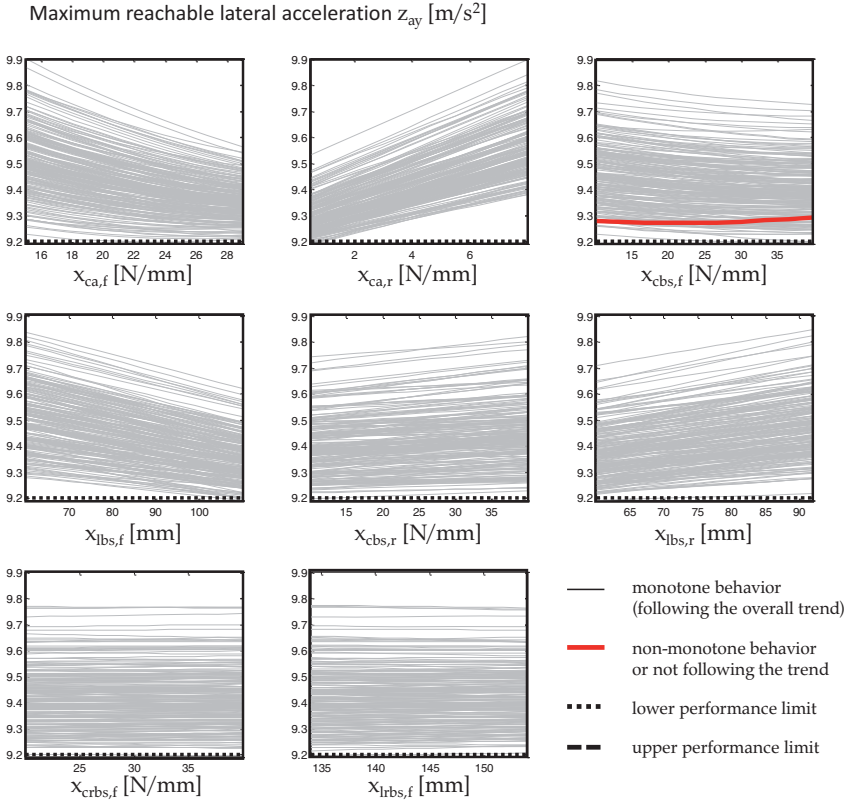


Figure B.4: One-dimensional sections of the twelve-dimensional input space with all other design variables and vehicle parameters assuming constant, randomly chosen values. The performance measure z_{ay} is plotted against each of the eight design variables.

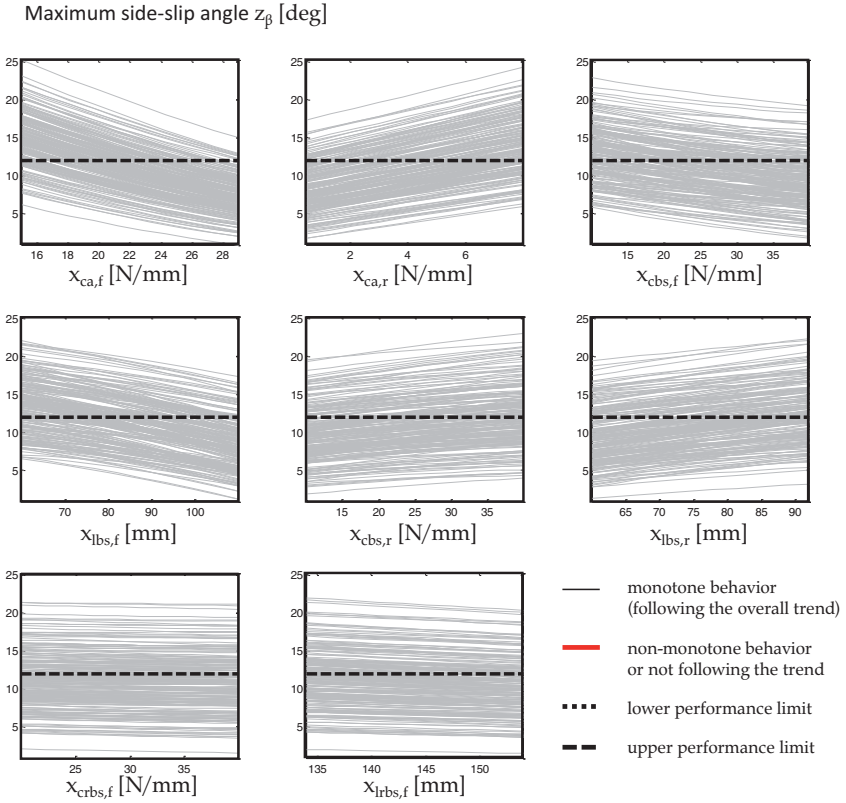


Figure B.5: One-dimensional sections of the twelve-dimensional input space with all other design variables and vehicle parameters assuming constant, randomly chosen values. The performance measure z_β is plotted against each of the eight design variables.

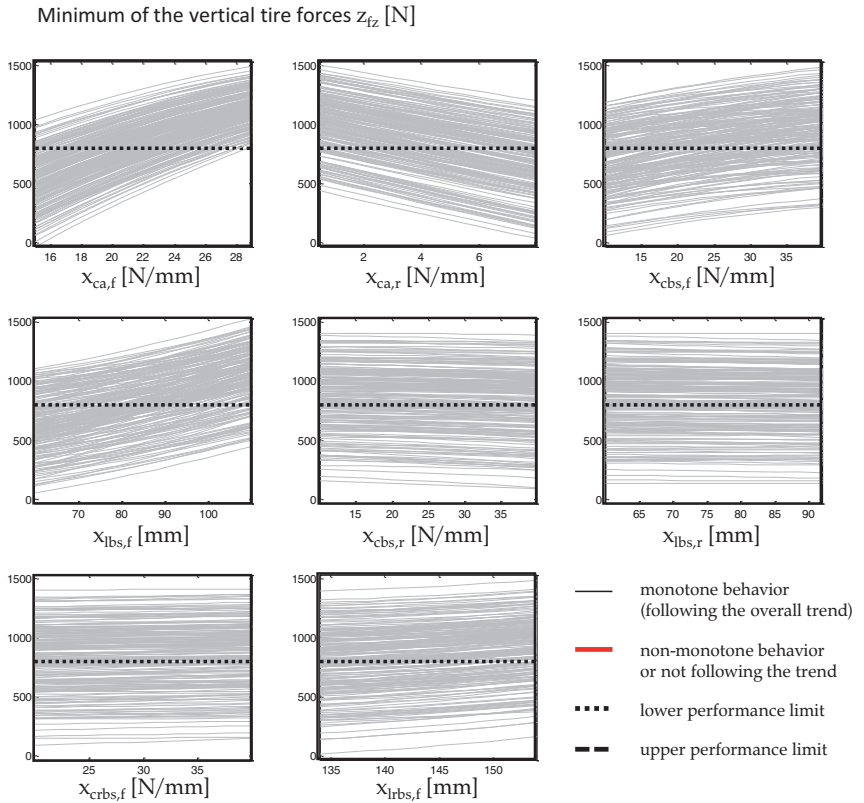


Figure B.6: One-dimensional sections of the twelve-dimensional input space with all other design variables and vehicle parameters assuming constant, randomly chosen values. The performance measure z_{fz} is plotted against each of the eight design variables.

B.4 Numerical details of the stochastic Solution Space algorithm

In the following the intervals as well as numerical details for five runs with the stochastic Solution Space algorithm for *Example One*, vehicle 1, are shown. The results are based on the linear model. Due to random Monte Carlo sampling the results are non-deterministic.

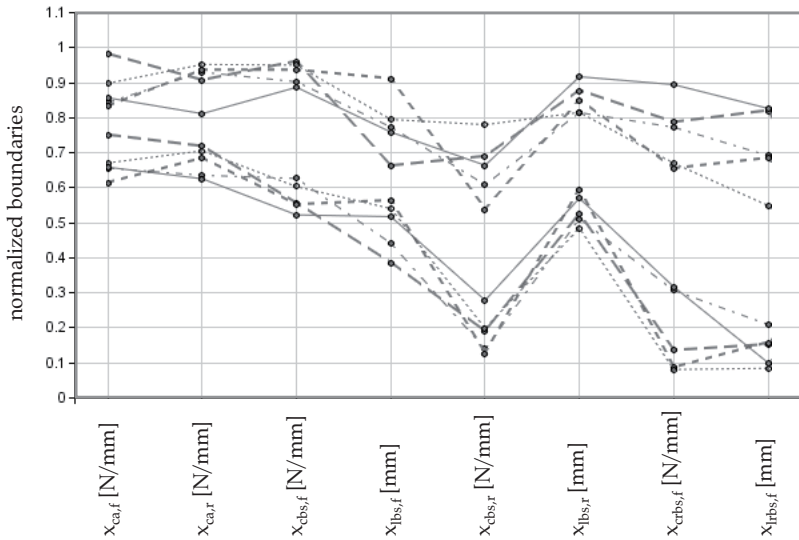


Figure B.7: Normalized intervals for five runs with the stochastic Solution Space algorithm. The results are based on the **linear model**, *Example One*, vehicle 1. Note that the lines between the lower and upper boundaries of the intervals are for visualization purposes only.

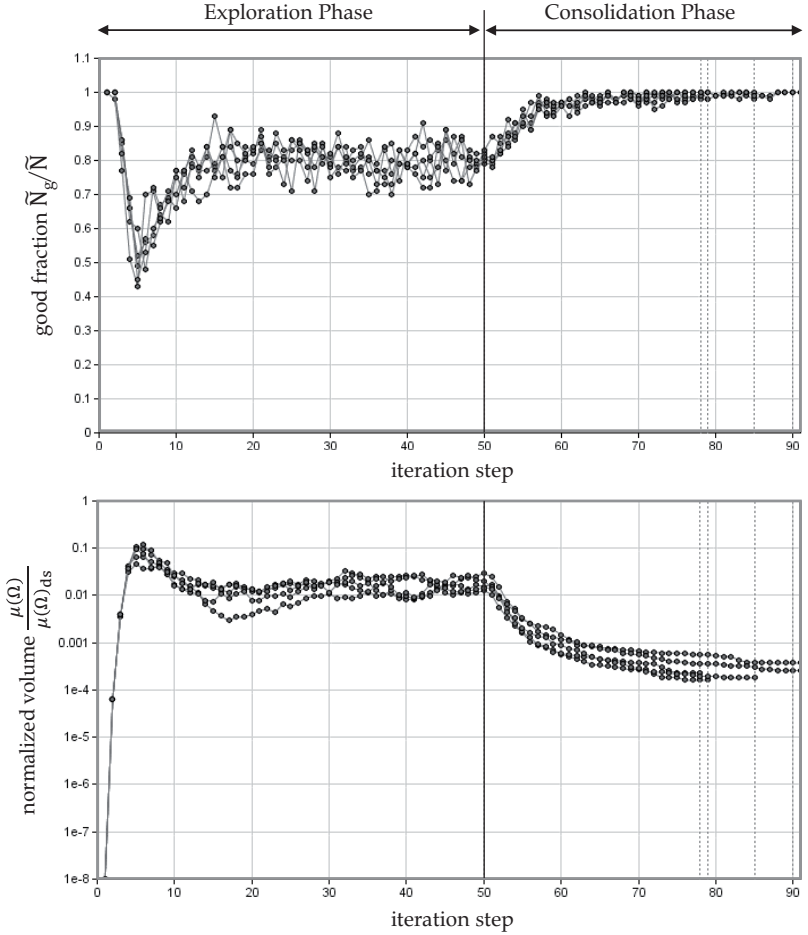


Figure B.8: Fraction of good designs within the box as well as normalized volume vs. iteration steps for five runs with the stochastic Solution Space algorithm. The results are based on the **linear model**, *Example One*, vehicle 1.

B.5 Computer specifications

For all computations, a standard computer with the following specifications was used:

Type - HP ZBook 15

Operating system - Microsoft Windows Enterprise, 64-bit

Processor - Intel(R) Core(TM) i7-4900MQ CPU @ 2.80GHz, 4 cores

Memory - 16 GB RAM

MATLAB® version - 2010b

Bibliography

- [1] *FMVSS 126, Electronic Stability Control Systems*. National Highway Traffic Safety Administration, USA, 2008.
- [2] A. Al-Ashaab, S. Howell, K. Usowicz, P. Hernando Anta, and A. Gorka. Set-based concurrent engineering model for automotive electronic/software systems development. In *Proceedings of the 19th CIRP Design Conference—Competitive Design*. Cranfield University Press, 2009.
- [3] G. Alefeld and G. Mayer. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121(1):421–464, 2000.
- [4] E. L. Allgower and P. H. Schmidt. Computing volumes of polyhedra. *Mathematics of Computation*, 46(173):171–174, 1986.
- [5] J. Antony. *Design of experiments for engineers and scientists*. Elsevier, 2014.
- [6] P. Armand, J. C. Gilbert, and S. Jan-Jégou. A feasible BFGS interior point algorithm for solving convex minimization problems. *SIAM Journal on Optimization*, 11(1):199–222, 2000.
- [7] C. Audet and J. E. Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002.
- [8] C. Audet and J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- [9] A. Basudhar and S. Missoum. Adaptive explicit decision functions for probabilistic design and optimization using support vector machines. *Computers & Structures*, 86(19):1904–1917, 2008.
- [10] M. Beer and M. Liescher. Designing robust structures—a nonlinear simulation based approach. *Computers & Structures*, 86(10):1102–1122, 2008.

- [11] J. I. Bernstein. *Design methods in the aerospace industry: looking for evidence of set-based practices*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1998.
- [12] H.-G. Beyer and B. Sendhoff. Robust optimization—a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33):3190–3218, 2007.
- [13] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
- [14] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
- [15] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.
- [16] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.
- [17] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4):377–409, 1993.
- [18] W. Chen and K. Lewis. Robust design approach for achieving flexibility in multidisciplinary design. *AIAA journal*, 37(8):982–989, 1999.
- [19] M. Daub. Konvexe Optimierung am Beispiel volumenmaximaler einbeschriebener Rechtecksmengen. Master’s thesis, Universität Konstanz, Germany, 2017.
- [20] O. Deiser. *Einführung in die Mengenlehre*, volume 3. Springer, 2002.
- [21] A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31(2):105–112, 2009.
- [22] M. Eichstetter. *Design of Vehicle System Dynamics using Solution Spaces*. PhD thesis, Technische Universität Berlin, Germany, 2017.
- [23] M. Eichstetter, S. Müller, and M. Zimmermann. Product Family Design With Solution Spaces. *ASME - Journal of Mechanical Design*, 137(12):121401, 2015.
- [24] M. Eichstetter, C. Redeker, P. Kvasnicka, S. Müller, and M. Zimmermann. Solution Spaces for Damper Design in Vehicle Dynamics. *5th International Munich Chassis Symposium, Munich, Germany*, 2014.

- [25] I. Enevoldsen and J. D. Sørensen. Reliability-based optimization in structural engineering. *Structural Safety*, 15(3):169–196, 1994.
- [26] S. Erschen, F. Duddeck, M. Gerdts, and M. Zimmermann. On the optimal decomposition of high-dimensional solution spaces of complex systems. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 4(2), doi: 10.1115/1.4037485, 2017.
- [27] S. Erschen, F. Duddeck, and M. Zimmermann. Robust Design using classical optimization. *PAMM*, 15(1):565–566, 2015.
- [28] J. Fender. *Solution Spaces for Vehicle Crash Design*. PhD thesis, Technical University of Munich, Germany, 2014.
- [29] J. Fender, F. Duddeck, and M. Zimmermann. Direct computation of solution spaces. *Structural and Multidisciplinary Optimization*, 55(5):1787–1796, 2017.
- [30] J. Fender, L. Graff, H. Harbrecht, and M. Zimmermann. Identifying Key Parameters for Design Improvement in High-Dimensional Systems With Uncertainty. *ASME - Journal of Mechanical Design*, 136(4):041007, 2014.
- [31] W. W. Finch and A. C. Ward. Quantified Relations: A Class of Predicate Logic Design Constraints Among Sets of manufacturing, Operating, and Other Variations. In *Proceedings of the 8th International Conference on Design Theory and Methodology*, Irvine, California, USA, 1996.
- [32] W. W. Finch and A. C. Ward. A set-based system for eliminating infeasible designs in engineering problems dominated by uncertainty. In *Proceedings of the 1997 ASME Design Engineering Technical Conferences, Sacramento, CA, Paper No. DETC97/DTM-3886*, USA, 1997.
- [33] J. Fox. *Applied regression analysis and generalized linear models*. Sage Publications, 2015.
- [34] G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 32–40, Chicago, USA, 2005. ACM.
- [35] M. Götz, M. Liebscher, and W. Graf. Efficient detection of permissible design spaces in an early design stage. In *Proceedings of the 11th LS-Dyna Forum*, pages 9–10, Ulm, Germany, 2012.
- [36] L. Graff. *A stochastic algorithm for the identification of solution spaces in high-dimensional design spaces*. PhD thesis, University of Basel, Switzerland, 2013.

- [37] L. Graff, H. Harbrecht, and M. Zimmermann. On the computation of solution spaces in high dimensions. *Structural and Multidisciplinary Optimization*, 54(4):811–829, 2016.
- [38] R. Haberfellner and W. F. Daenzer. *Systems Engineering: Grundlagen und Anwendung*. Orell Füssli, 2012.
- [39] M. Hanss. The transformation method for the simulation and analysis of systems with uncertain parameters. *Fuzzy Sets and Systems*, 130(3):277–289, 2002.
- [40] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2002.
- [41] C. Jianjiang, X. Renbin, and Z. Yifang. A response surface based hierarchical approach to multidisciplinary robust optimization design. *The International Journal of Advanced Manufacturing Technology*, 26(4):301–309, 2005.
- [42] E. Keogh and A. Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer, 2011.
- [43] N. Kieft. *Evaluation of different design space methods for Analyzing Combustion Engine Operation Limits*. PhD thesis, Universit at Leiden, The Netherlands, 2014.
- [44] P. Kvasnicka, G. Prokop, M. Dorle, A. Rettinger, and H. Stahl. Durchg angige Simulationsumgebung zur Entwicklung und Absicherung von Fahrdynamischen Regelsystemen. *VDI Berichte*, 1967(1):387, 2006.
- [45] M. Lehar and M. Zimmermann. An inexpensive estimate of failure probability for high-dimensional systems with uncertainty. *Structural Safety*, 36:32–38, 2012.
- [46] J. K. Liker, D. K. Sobek, A. C. Ward, and J. J. Cristiano. Involving suppliers in product development in the United States and Japan: Evidence for set-based concurrent engineering. *IEEE Transactions on Engineering Management*, 43(2):165–178, 1996.
- [47] S. Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.
- [48] B. M oller and M. Beer. *Fuzzy randomness: uncertainty in civil engineering and computational mechanics*. Springer Science & Business Media, 2013.
- [49] D. C. Montgomery. *Design and analysis of experiments*. John Wiley & Sons, 2008.
- [50] M. M nster, M. Lehner, D. Rixen, and M. Zimmermann. Vehicle steering design using solution spaces for decoupled dynamical subsystems. In *Proceedings of the*

- 26th conference on Noise and Vibration Engineering (ISMA2014)*, volume 26, pages 279–288, Leuven, Belgium, 2014.
- [51] Y.-E. Nahm and H. Ishikawa. Novel space-based design methodology for preliminary engineering design. *The International Journal of Advanced Manufacturing Technology*, 28(11-12):1056–1070, 2006.
- [52] N.N. *Passenger cars – Steady-state circular driving behaviour – Open-loop test methods*. International Organization for Standardization, 2012.
- [53] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [54] H. Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.
- [55] M. Padulo. *Computational engineering design under uncertainty: an aircraft conceptual design perspective*. PhD thesis, Cranfield University, UK, 2009.
- [56] J. H. Panchal, M. G. Fernández, J. K. Allen, C. J. Paredis, and F. Mistree. An interval-based focalization method for decision-making in decentralized, multi-functional design. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 413–426. American Society of Mechanical Engineers, 2005.
- [57] D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [58] B. Prasat. *Concurrent Engineering Fundamentals Volume 1: Integrated Product and Process Organization*. Prentice-Hall, 1997.
- [59] B. Prasat. *Concurrent Engineering Fundamentals Volume 2: Integrated Product Development*. Prentice-Hall, 1997.
- [60] T. Rauch. Berechnung von Lösungsräumen mit klassischer Optimierung in der Grundaulegung Fahrdynamik. Master’s thesis, Universität der Bundeswehr, Munich, Germany, 2015.
- [61] S. J. Rekuc, J. M. Aughenbaugh, M. Bruns, and C. J. Paredis. Eliminating design alternatives based on imprecise information. Technical report.
- [62] C. M. Rocco, J. A. Moreno, and N. Carrasquero. Robust design using a hybrid-cellular-evolutionary and interval-arithmetic approach: a reliability application. *Reliability Engineering & System Safety*, 79(2):149–159, 2003.

- [63] C. Roeski. *Eine Methode zur simulationsbasierten Grundausslegung von PKW-Fahrwerken mit Vertiefung der Betrachtung zum Fahrkomfort*. PhD thesis, Technical University of Munich, Germany, 2012.
- [64] J. H. Saleh, G. Mark, and N. C. Jordan. Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems. *Journal of Engineering Design*, 20(3):307–323, 2009.
- [65] R. Schindler. *Set Theory - Exploring Independence and Truth*. Springer, 2014.
- [66] T. W. Simpson. Product platform design and customization: Status and promise. *Artificial intelligence for engineering design, analysis and manufacturing*, 18(1):3–20, 2004.
- [67] T. W. Simpson, J. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: survey and recommendations. *Engineering with computers*, 17(2):129–150, 2001.
- [68] D. J. Singer, N. Doerry, and M. E. Buckley. What Is Set-Based Design? *Naval Engineers Journal*, 121(4):31–43, 2009.
- [69] D. K. Sobek II. A set-based model of design. *Mechanical Engineering*, 118(7):78, 1996.
- [70] D. K. Sobek II, A. C. Ward, and J. K. Liker. Toyota’s principles of set-based concurrent engineering. *MIT Sloan Management Review*, 40(2):67, 1999.
- [71] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical programming*, 107(3):391–408, 2006.
- [72] A. Ward, J. K. Liker, J. J. Cristiano, and D. K. Sobek. The second Toyota paradox: How delaying decisions can make better cars faster. *MIT Sloan Management Review*, 36(3):43, 1995.
- [73] A. C. Ward. A theory of quantitative inference applied to a mechanical design compiler. 1989.
- [74] A. C. Ward, J. K. Liker, D. K. Sobek, and J. J. Cristiano. Set-based concurrent engineering and Toyota. In *Proceedings of ASME Design Engineering Technical Conferences*, ASME, pages 79–90, 1994.
- [75] A. C. Ward and D. K. Sobek II. *Lean product and process development*. Lean Enterprise Institute, 2014.

-
- [76] S. S. Willaert, R. De Graaf, and S. Minderhoud. Collaborative engineering: A case study of Concurrent Engineering in a wider context. *Journal of Engineering and Technology Management*, 15(1):87–109, 1998.
- [77] J. Wimmeler and M. Zimmermann. Concurrent design of vehicle tires and axles. In *6th International Munich Chassis Symposium 2015*, pages 839–851. Springer, 2015.
- [78] M. Zimmermann and J. E. Hoessle. Computing solution spaces for robust design. *International Journal for Numerical Methods in Engineering*, 94(3):290–307, 2013.
- [79] M. Zimmermann, S. Königs, C. Niemeyer, J. Fender, C. Zeherbauer, R. Vitale, and M. Wahle. On the design of large systems subject to uncertainty. *Journal of Engineering Design*, 28(4):233–254, 2017.
- [80] M. Zimmermann and M. Wahle. Solution Spaces for Vehicle Concepts and Architectures. *24th Aachen Colloquium Automobile and Engine Technology, Aachen, Germany*, 2015.

Glossary

2d-space is a set of good designs, specified by a Cartesian product of two-dimensional sets, each assigned to a pair of design variables.

bad design is a design that doesn't satisfy all requirements.

box-shaped Solution Space is a set of good designs, specified by a Cartesian product of intervals for each design variable.

complete Solution Space is the entire set of good designs.

design is a system (here the overall vehicle) with specified properties of its subsystems and components.

design space is a set that comprises all possible combinations of design variable values.

design variable is a property of a component, e.g. torsional stiffness of an anti-roll bar, that must be specified within the development process.

good design is a design that satisfies all requirements.

inner box is a box that contains good designs only.

outer box is a box where all designs outside are bad.

performance function maps values of design variables onto a value of a performance measure.

performance measure is a quantitative measure of the vehicle's response, e.g. the roll angle of the vehicle while cornering.

Solution Space is a set of good designs, i.e. designs that satisfy all requirements. Note that the term *space* might be misleading referring to the mathematical definition of a space, however, here the term Solution Space is used in accordance with the literature.

Solution Space constraint is an inequality that must be satisfied by a design to be a good design.

system is an assembly of several subsystems and components, here the overall vehicle.

vehicle parameter is a property of the vehicle, e.g. overall mass of the vehicle.