Technische Universität München
TUM School of Engineering and Design

Informed Decomposition: Distributed design optimization of mechanical multi-component systems

Lukas Krischer

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. ir. Daniel J. Rixen

Prüfer*innen der Dissertation:

1. Prof. Dr. Markus Zimmermann
2. Prof. Dr.-Ing. Kai-Uwe Bletzinger

Die Dissertation wurde am 10.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 10.05.2023 angenommen.

Abstract Monolithic optimization of large systems with many interacting components can be a difficult task. Hence, distributed optimization architectures have been developed in which the monolithic optimization is decomposed into a set of smaller optimization subproblems. However, most of the distributed architectures are not fully separable and require a coordination strategy among the subproblems.

In this work, a new distributed optimization architecture is proposed that does not require coordination. The proposed architecture consists of two parts: (1) a system optimization that assigns stiffness requirements to components that are approximately feasible and mass-optimal, based on pre-trained meta models, and (2) component optimizations that can be solved independently and in parallel. Because the creation of meta models for the system optimization is expensive, an offline database consisting of multiple meta models valid for a variety of design problems, is created. The required training data is computed using a newly developed active-learning undersampling strategy that enables an efficient sampling process while also providing a well-balanced dataset.

The validity of the proposed approach is demonstrated by minimizing the mass of a two-component system subject to a displacement requirement. With increasing complexity, the quality of results slightly deteriorates from 0.40% to 3.91% to a maximum mass deviation of 8.18% compared to a monolithic optimization. Next, a four-component system is investigated to path the way towards practical application. For this purpose, the general applicability of the offline database for components of different geometrical dimensions is investigated, a computational time comparison is performed, and finally the approach is applied to design a low-cost lightweight robot. The results obtained for the robot application deviate from the benchmark result of monolithic optimization by 12.9%.

Keywords Distributed Optimization · Topology Optimization · Machine Learning · Lightweight Design

TABLE OF CONTENTS

1 Introduction

1.1 Motivation for distributed design optimization

A useful industry benchmark for vehicles is that a 10% reduction in weight results in a 6% improvement of fuel consumption of a combustion-engine automotive or in a 14% increase in range for electric vehicles (Robinson et al., 2019). This not only illustrates the potential of weight reduction with respect to environmental protection, but also the economical aspect. Weight reduction and, thus, lightweight design therefore plays a significant role in modern product development processes. Lightweight design can be defined as the minimization of weight while satisfying all system requirements. Apart from the automotive industry, this applies in particular to all non-stationary applications, such as robots, ships, aircraft and spacecraft, but also to stationary products, such as bridges in civil engineering.

To design the lightest possible system, structural design optimization schemes can be utilized. Structural design optimization is a mathematical design method to find the optimal or best design within the available means (Papalambros & Wilde, 2018; Martins & Ning, 2022). In general, global optimality can rarely be proven and rather optimized designs are achieved with respect to a given benchmark (Sigmund, 2011). Since the idea for analytical calculation of mass-minimal truss structures by Michell (1904), design optimization methods have undergone numerous improvements and have been applied to many different problems. While various different approaches for structural optimization exist, topology optimization has become a frequently used design approach, especially in early stages of product development processes. Most developments are based on the works of Bendsøe & Kikuchi (1988) and Bendsøe & Sigmund (2004) and many other methods have evolved from there. The amount of research that has been done on this topic is extensive and continues to grow, as can be seen from the number of literature reviews published in recent years, e.g., by Rozvany (2009), Sigmund & Maute (2013), Deaton & Grandhi (2014) or L. Wang et al. (2021).

However, the holistic or monolithic design of large systems with many interacting components can be a difficult task. First, from a product development perspective, the development process itself requires decomposition (Sobieszczanski-Sobieski & Haftka, 1997). In classical top-down development processes, requirements are first formulated at the system level and then broken down, and passed on to lower levels and finally to the respective departments (Forsberg & Mooz, 1991). The tasks are then ideally solved by separate engineering groups. This hierarchical process is particularly advantageous from a designer's perspective, who can use specialized analysis and design tools to work on parts rather than the entire system (Eckert & Clarkson, 2005; Tosserams et al., 2009). Second, from a computational perspective, the sheer size of the system can make monolithic optimization prohibitively expensive, requiring decomposition to reduce the size of the problem and the computational time (Martins & Lambe, 2013).

For both views, decomposition is therefore often preferred over a monolithic architecture. In the context of design optimization, distributed design optimization architectures have been developed to reduce computational time and resemble this distributed development process (Martins & Ning, 2022). Distributed optimization architectures decompose a given optimization problem into smaller optimization subproblems that allow individual design by separate groups (Martins & Lambe, 2013). In recent years, many different distributed architectures have been developed and successfully applied to a variety of different use cases. Among others, collaborative optimization by Braun (1996), analytical target cascading by H. M. Kim et al. (2003), BLISS-2000 by Sobieszczanski-Sobieski et al. (2000, 2003) or quasiseparable decomposition by Haftka & Watson (2005).

1.2 Lightweight design of mechanical multi-component systems

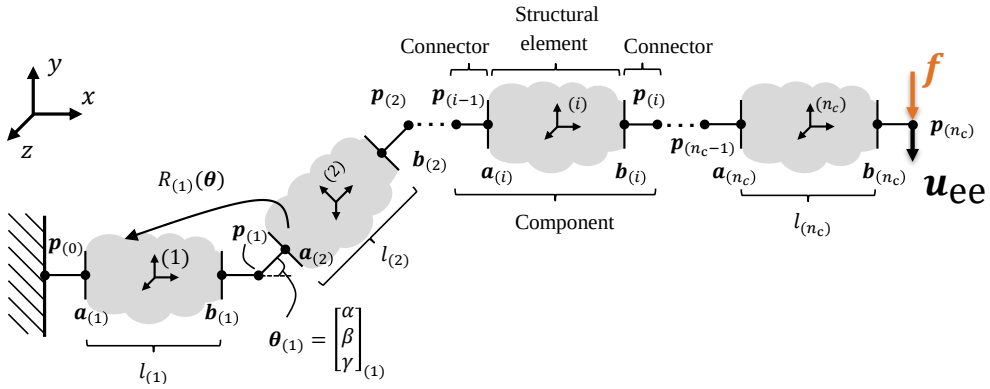


Figure 1.1: A generic serial mechanical multi-component system clamped on the left side of the first component and subject to a load $\boldsymbol{f}$ on the right end of the system resulting in a translational end effector displacement $\boldsymbol{u}_{ee}$

In this work, the lightweight design of mechanical multi-component systems is to be investigated. A mechanical multi-component system is hereby defined as a collection of mechanical components that interact with each other and perform specified functions. According to this definition, a large number of industrial products can be considered as mechanical multi-component systems. For research purpose, the following investigations are restricted on serial systems, yet an extension to parallel systems is possible without any changes within the proposed approach.

A generic serial mechanical multi-component system, Fig. 1.1, has n_c components that can be oriented with respect to each other by prescribed rotations $\boldsymbol{\theta}_{(i)} = [\alpha, \beta, \gamma]^\top_{(i)}$ at the joint positions $\boldsymbol{p}_{(i)}$, where each set of rotations yields a specific system pose. Each component i consists of a structural element and two rigid connectors on both sides. The structural elements of length $l_{(i)}$ are modeled as linear elastic and have two mechanical interfaces $\boldsymbol{a}_{(i)}$ and $\boldsymbol{b}_{(i)}$ with n_dof=6 degrees of freedom. The first component is clamped on the left side $\boldsymbol{p}_{(0)}$ and a static payload $\boldsymbol{f}$ is applied on the right side of the last component of the system $\boldsymbol{p}_{(n_\mathrm{c})}$. The combination of pose and acting total system load $\boldsymbol{f}_\mathrm{s}$ is called load case and the system can be investigated with respect to multiple load cases n_p.

The requirement on the system stiffness is:

> The system must sustain a system load $\boldsymbol{f}_\mathrm{s}$ with a maximum translational end effector displacement of u_max for a specified set of rotations $\boldsymbol{\theta}_{(i)}$.

Fig. 1.2 illustrates the dependencies between all relevant quantities that are needed to solve the given design problem. The detailed design variables $\boldsymbol{x}_{(i)}$ include all design details for component i. For a given material, $\boldsymbol{x}_{(i)}$ determines the detailed stiffness matrix $\boldsymbol{K}_{\mathrm{d},(i)}$, including all degrees of freedom of each structural element, and subsequently the interface stiffness matrix $\boldsymbol{K}_{(i)} \in \mathbb{R}^{12\times12}$ that defines the component's elastic behavior with respect to the two interfaces $\boldsymbol{a}_{(i)}$ and $\boldsymbol{b}_{(i)}$. The components are then assembled to the system stiffness matrix $\boldsymbol{K}_\mathrm{s} = \mathrm{A}_{i=1}^{n_\mathrm{c}} \boldsymbol{K}_{(i)}$. Under a given system load vector $\boldsymbol{f}_\mathrm{s}$, the system deforms, resulting in the general system displacements $\boldsymbol{d}_\mathrm{s}$. Since the requirement is only on the translational part of the deformation $\boldsymbol{d}_\mathrm{s}$, the system stiffness is measured as the inverse of

$$u = ||\boldsymbol{u}_\mathrm{ee}||_2, \tag{1.1}$$

whereas $\boldsymbol{u}_\mathrm{ee} \in \boldsymbol{d}_\mathrm{s}$ and contains the translational displacements at the end effector of the system. Similarly, the detailed design variables $\boldsymbol{x}_{(i)}$ also define the mass $m_{(i)}$ of each component and consequently the

system mass

$$m = m_{\mathrm{s}} = \sum_{i=1}^{n_{\mathrm{c}}} m_{(i)}. \tag{1.2}$$

The design problem can be hierarchically organized into three levels:

(I) system level: $\boldsymbol{z} = [m, u]$,

(II) component-performance level: $\boldsymbol{y}_{(i)} = [m_{(i)}, \boldsymbol{K}_{(i)}]$,

(III) component-detail level: $\boldsymbol{x}_{(i)}$.

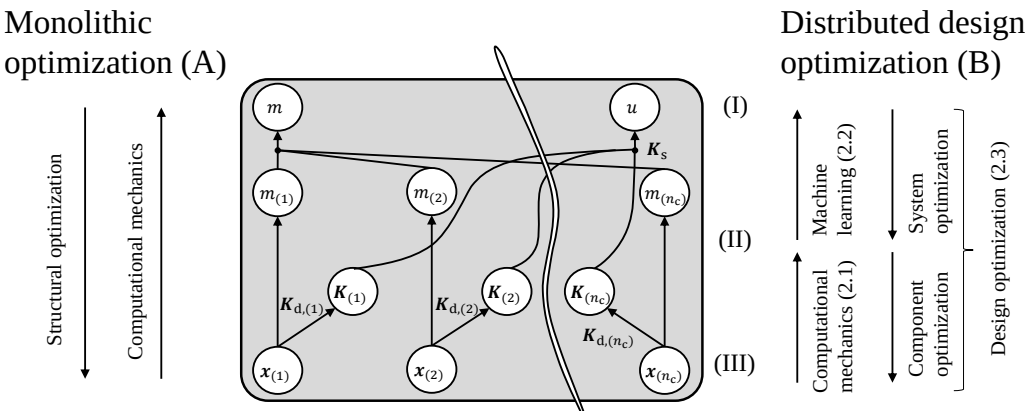


Figure 1.2: Dependencies between all relevant quantities on the system level (I), component-performance level (II), and component-detail level (III) for monolithic optimization (A) and a distributed design optimization approach (B)

In order to solve the design problem, it can be cast into a monolithic structural optimization problem (A)

$$\begin{aligned} & \min_{\boldsymbol{x}_{(i)}} \textstyle\sum_{i=1}^{n_{\mathrm{c}}} m_{(i)}\left(\boldsymbol{x}_{(i)}\right), \\ \text{s. t.:} \quad & u_c\left(\boldsymbol{x}_{(i)}\right) - u_{\max} \leq 0, \quad \text{for } c{=}1, ..., n_{\mathrm{p}}, \\ & \boldsymbol{x}_{\mathrm{lb}} \leq \boldsymbol{x}_{(i)} \leq \boldsymbol{x}_{\mathrm{ub}}, \quad \text{for } i{=}1, ..., n_{\mathrm{c}}. \end{aligned} \tag{1.3}$$

The optimization problem (1.3) has some particular properties. There are no shared design variables $\boldsymbol{x}_{(0)}$ between the components of the system, meaning each component possesses only its own design variables $\boldsymbol{x}_{(i)}$. The objective function $m{=}\sum_{i=1}^{n_{\mathrm{c}}} m_{(i)}$ is separable, i.e., it can be expressed as a sum of functions, each of which depend only on the corresponding local design variables $\boldsymbol{x}_{(i)}$. On the other hand, the constraint function $g(\boldsymbol{x}){=}u_c(\boldsymbol{x}) - u_{\max}$ depends on all design variables $\boldsymbol{x}{=}[\boldsymbol{x}_{(1)}, ..., \boldsymbol{x}_{(n_{\mathrm{c}})}]$. If the constraints $\boldsymbol{g}(\boldsymbol{x}){\leq}0$ did not exist, this optimization problem could be simply decomposed into n_{c} independent optimization subproblems. In literature, this kind of monolithic design optimization problem is called *complicating constraints problem* (Conejo et al., 2002).

For distributed architectures, the monolithic optimization problem (1.3) therefore needs to be decomposed. The fundamentals needed for a new distributed optimization architecture (B) are explained in the following Chapter 2, including the required methods of computational mechanics in Section 2.1, machine learning in Section 2.2, and design optimization methods in Section 2.3.

1.3 Problem description and research objectives

In Section 1.1 it was shown that the lightweight design of complex systems in a monolithic manner, such as the serial mechanical multi-component system problem introduced in Section 1.2, motivates the development of distributed optimization architectures. The two main motivations are that product development in industry, involving many parties, requires a distributed process and that decomposing the problem can reduce computational time.

However, in practice, distributed optimization architectures are not able to sufficiently solve these two problems, and thus have not been widely used in industry (Martins & Ning, 2022). A major reason for this is that the decomposed optimization problems of most distributed architectures are not fully separable and require a coordination strategy to maintain consistency between the shared variables of the system and the subproblems. In an industrial context, this coordination would need an optimization architecture connecting all involved departments, which is difficult to realize, because different engineering groups may use different design tools. In addition, when using such a coordination strategy, there is a risk that the actual coordination cost will exceed the cost of the original optimization problem, which is referred to as coordination overhead in design optimization (Alexandrov & Hussaini, 1997; Martins & Lambe, 2013; Tosserams et al., 2009). Unless a problem has a particular structure, there is no distributed architecture that converges as fast as a monolithic one (Martins & Ning, 2022). Sometimes distributed optimization architectures even fail to converge to a solution that is consistent and/or feasible, i.e., satisfies all posed requirements.

Therefore, a decomposition scheme without any need for coordination is advantageous. In the context of distributed optimization architectures, we define the term *decoupling* as a decomposition between system and component level, and a decomposition between different components without the need for coordination after the decomposition. Thus, the implementation and adoption of decoupled distributed optimization architectures in industry is simplified in comparison to classical distributed architectures. This type of decoupling, for example, was introduced by Zimmermann & von Hoessle (2013) and embedded in a design procedure in Zimmermann et al. (2017). However, the procedure differs from classical design optimization methods in the way that it is not based on point solutions, but so-called solution spaces, which represent sets of good, i.e., feasible, designs. Regardless of the optimization objectives, decoupled architectures generally run into the risk of

1. accidentally ruling out the optimal but initially unknown solutions on the system level and
2. committing to a *physically infeasible* design, i.e., a design that cannot be realized on a detail level afterwards.

These are two classical problems from top-down design tasks that specify certain performance measures a priori. One way to address this is to transfer a priori information from the component level to the system level using meta models, i.e., mathematical surrogates that are trained before performing a decoupling (Hou & Jiao, 2020). Meta models have a long history in systems design because they can reduce the high computational cost of large-size computer models (Viana et al., 2014). Papadrakakis et al. (1998) were one of the first authors that successfully applied meta models into a structural optimization problem, yet for size and shape optimization only. In the field of topology optimization, the combination of high-fidelity solutions of finite element models together with a large number of design variables lead to high computational cost for the design optimization process. Hence, meta models have recently been used to speed up the optimization process of these large-scale optimization problems, see, for example, the reviews of Mukherjee et al. (2021), Ramu et al. (2022) or Woldseth et al. (2022).

Here, the deformation energy, referred to as compliance in the context of topology optimization, is often utilized as a performance measure. Since the compliance is a load dependent quantity, corresponding mechanical detailed designs are only valid for the respective load case. For example, B. J. Kim et al.

(2016) carried out a dynamic system optimization with a decomposition of the structure based on a regression model that maps mass onto compliance for equivalent static loads of different time steps. Oh et al. (2019) utilized deep learning techniques for estimating topologies with respect to minimal compliance for given normal and shear load cases of a vehicle wheel. The theoretical extension of this idea is that by using a sufficiently broad dataset that spans variations in loads, boundary conditions, material models, objective function, and design domains, one can train a regressor to construct a general mapping from the input conditions of a given problem onto its corresponding optimal topology. A variety of papers have been recently published on this idea, e.g., Ulu et al. (2015), Lei et al. (2019), Yu et al. (2019) or Sosnovik & Oseledets (2019). To take these variations into account, extensive sampling procedures are needed. In addition, compliance often does not accurately represent stiffness-related quantities of interest, such as maximum displacement. Using compliance here would impose an unnecessarily strict constraint on a problem. However, similar to compliance, the displacement of a structure is load dependent and therefore requires extensive sampling, too.

In order to reduce the amount of sample data needed, it would be beneficial to have quantities that represent inherent component characteristics and are thus independent of changing boundary conditions. For linear analyses, a stiffness matrix incorporates all information about the geometrical and physical constitution of mechanical components and is therefore load-independent. Moreover, it can be directly related to a displacement requirement for a given load case and contains all relevant information regarding the deformation behavior. Hence, stiffness modeling plays an important role, e.g., in the design of robots, where multiple poses and loads are often considered. To reduce computational cost, meta models are sometimes used here to investigate the elastic behavior. For example, X. Wang et al. (2019) performed a stiffness matrix optimization of a serial robot based on a parameterization of a topology, using a linear regression for the stiffness estimation of the components with respect to the mass. Parameterization here refers to more general geometrical changes through some few representative design variables called parameters, rather than detailed changes within the topology. Furthermore, M. Wang et al. (2022) topology optimized a parallel robot using a stiffness matrix-mass meta model for different geometrical dimensions trained by an orthogonal design of experiment. In the context of multi-scale optimization, the expensive connection of macro- and microscale of a structure is sometimes replaced by meta models to reduce computational cost. These meta models estimate the elastic behavior of the microstructure in terms of internal energy or the constitutive tensor, which can be related to the stiffness matrix of a mechanical body. For instance, a substructuring technique for hierarchical lattice structures was used in Z. Wu et al. (2019) to estimate the mass and stiffness matrix of microstructures. For this purpose, a regression model based on orthogonal decomposition was built, where microstructure properties are determined by samples of parameterized unit cells. White et al. (2019) proposed a neural network to provide a mapping from parameterized microstructures to their elastic material properties as elastic stiffness coefficients. Further, L. Wang et al. (2021) performed an optimization considering meta models of different parameterized microstructures in terms of material properties, such as stiffness matrix or thermal conductivity. However, due to the given parameterization of the geometry, the design freedom in the presented approaches is limited and thus one does not exploit the full lightweight design potential.

In Xia & Breitkopf (2015), topology optimization without parameterization was performed to predict the effective strain energy density as well as the constitutive tensor of microstructures. Therefore, an offline database based on tensor decomposition was created for a continuous representation of topology optimized microstructures. Following the idea of an offline database, Ferrer et al. (2016) created a material catalog of microscale optimized topologies with respect to the constitutive tensor and compliance. The offline database is computed once in an offline process and can then be consulted as many times as needed in the online design process. In addition, Kollmann et al. (2020) has used equivalent load cases to determine the stiffness matrix of topology-optimized microstructures, training a regression model for multi-scale optimization. However, this regression model does not work directly with a stiffness matrix, but with either the bulk modulus, shear modulus or Poisson's ratio. Finally,

Yilin et al. (2021) trained convolutional neural networks for nonparametric microstructures using a voxel-based homogenization approach to calculate the effective elasticity tensor and its gradients. However, the microstructures were not optimized but created from a dataset for different topologies and volume fractions. In summary, to the author's knowledge, an application of meta models for topology-optimized structures that are not parameterized and work directly with stiffness matrices as input is still lacking.

Besides the risk of excluding optimal designs, ensuring physical feasibility is also a relevant and difficult challenge for decoupled top-down design approaches. Theoretically, all positive (semi-)definite stiffness matrices can be realized by a mechanical design. Milton & Cherkaev (1995) have shown that any given positive definite elasticity tensor satisfying the necessary symmetry conditions can be realized with a two-phase composite consisting of a sufficiently compliant isotropic phase and a sufficiently rigid isotropic phase configured in a suitable microstructure. Huang & Schimmels (1998) and Huang & Schimmels (2000) have shown that arbitrary spatial stiffness matrices can be realized with a set of so-called screw springs. Based on these findings, one could simply explicitly dismiss designs that are not positive definite to ensure feasibility for the system. This has been done, for instance, by L. Wang et al. (2020), who considered positive definiteness explicitly as an inequality constraint in a multi-scale optimization problem. However, in the design of continuous structural components, external influences, such as a limited geometrical design domain, available materials, and minimum member size further restrict the feasible design space of positive definite stiffness matrices (Milton et al., 2017; J. Wu et al., 2021). To avoid committing to an infeasible system level design that cannot be realized, a more appropriate feasibility constraint must be imposed, with design space limits that are not explicitly known yet. Existing data can be used to approximate the feasible region using machine learning classifiers. In a more general context, classifiers have frequently been used in top-down design to ensure feasibility, e.g., for analog circuit design (Ding & Vemur, 2005; Boolchandani et al., 2011), air conditioners (Jeong et al., 2012) or the physically feasible workspace of a robot (Kulick et al., 2013). In mechanical material design, Jung et al. (2019) modeled feasibility constraints via a support vector machine to perform optimization for inverse material design. Qiu et al. (2021) developed a deep learning-based design strategy for efficient and effective selection of fiber materials and stacking orientations of composites using a classifier that estimates physical feasibility based on a given database. Regenwetter & Ahmed (2022) developed a metric that considered physical feasibility, expressed as geometrical compatibility, for inverse design tasks, such as a bicycle frame. However, no previous work could be identified for direct and explicit classification of stiffness matrices.

In the area of distributed design optimization, some classical architectures also explicitly use meta models or at least recommend their use. In BLISS-2000, for example, the expensive training process of regression models is an explicit part of the optimization architecture itself in order to provide subproblem information to the system level during optimization (Sobieszczanski-Sobieski et al., 2000, 2003). In contrast, quasiseparable decomposition recommends the use of pre-trained regression models so that the training process itself is not part of the actual architecture. While the original version of the quasiseparable decomposition recommends only the use of regression models, some versions rely entirely on pre-trained models, but these do not include classification for physical feasibility and are also not intended for reuse (Haftka & Watson, 2005; B. Liu et al., 2004). An *offline database* for both, pre-trained regression and classification models valid for a wide range of design problems could therefore help to avoid the costly training process, so that training is only required when no suitable models are available. Such offline databases are used, for example, in some of the multi-scale optimizations presented previously, such as in Xia & Breitkopf (2015) or Ferrer et al. (2016), but an application to distributed optimization architectures has not yet been discovered in literature.

In conclusion, the motivation of the Section 1.1 and the preceding problem description of the state of the art contributions of this Section 1.3 have shown the opportunities and obstacles faced in the design of complex systems using design optimization and, in particular, distributed design optimization.

The goal of this work is now to:

> Develop a new **hierarchical** and **decoupled** distributed optimization architecture (B) for the lightweight design of mechanical multi-component systems, Fig. 1.1, by decoupling the original monolithic optimization problem (A) of (1.3) using meta models for physical feasibility and optimality as part of an offline database.

The main research objectives are:

1. Set up a surrogate-based **system optimization** problem between hierarchy level (I) and (II) that decouples the optimization problem by assigning stiffness requirements that are approximately physically feasible and mass-optimal. Information about physical feasibility and optimality should be provided by meta models during the system optimization. Decoupling is to be done according to the physical components, which is an object-based partitioning.
2. Set up a **component optimization** formulation between hierarchy level (II) and (III), that can be solved independently of each other and therefore also in parallel, while still ensuring that the requirement on system stiffness $u \leq u_{\max}$ is satisfied.
3. Create an **offline database** of available meta models that can be used for the system optimization without the necessity of training a meta model for every single design task. The meta models estimate physical feasibility and optimality with respect to mass based on the interface stiffness matrix of each component.

1.4 Structure of the thesis

The organization of this thesis is illustrated in Fig. 1.3. After the introduction of this chapter, Chapter 2 explains the fundamentals needed to build a decoupled optimization architecture for mechanical multi-component systems, see also Fig 1.2. Section 2.1 presents the methods of computational mechanics necessary to build the bottom-up mapping between level (III) and (II), while Section 2.2 introduces the basics of machine learning to create meta models for the connection between (II) and (I). Utilizing those bottom-up mappings, design optimization methods are presented in Section 2.3. Surrogate-based optimization between level (I) and (II) is explained in Section 2.3.1 and topology optimization between level (II) and (III) in Section 2.3.2. The subsequent Chapter 3 presents the current state of the art related to distributed design optimization methods in detail and outlines the challenges and research gaps of the existing methods. Based on the fundamentals and already existing distributed architectures, Chapter 4 presents the proposed approach denoted as *Informed Decomposition* (B). In Chapter 5, a two-component system is investigated to verify the validity of the proposed architecture in terms of physical feasibility and optimality with respect to mass for the design problem class (P1). Subsequently, in Chapter 6, the approach is further analyzed to show the way towards a practical application of the developed method by investigating the design problem class (P2). A four-component system is therefore analyzed. The objectives of the second investigation are the verification of the general applicability of the offline database for components with different geometrical dimensions, the investigation of computational time, and finally the structural design of a low-cost lightweight robot for a pick-and-place task. Chapter 7 provides a general discussion on the results and shows both advantages and disadvantages of the developed approach. Finally, Chapter 8 concludes with a summary and an outlook on possible improvements and extensions of the proposed Informed Decomposition.

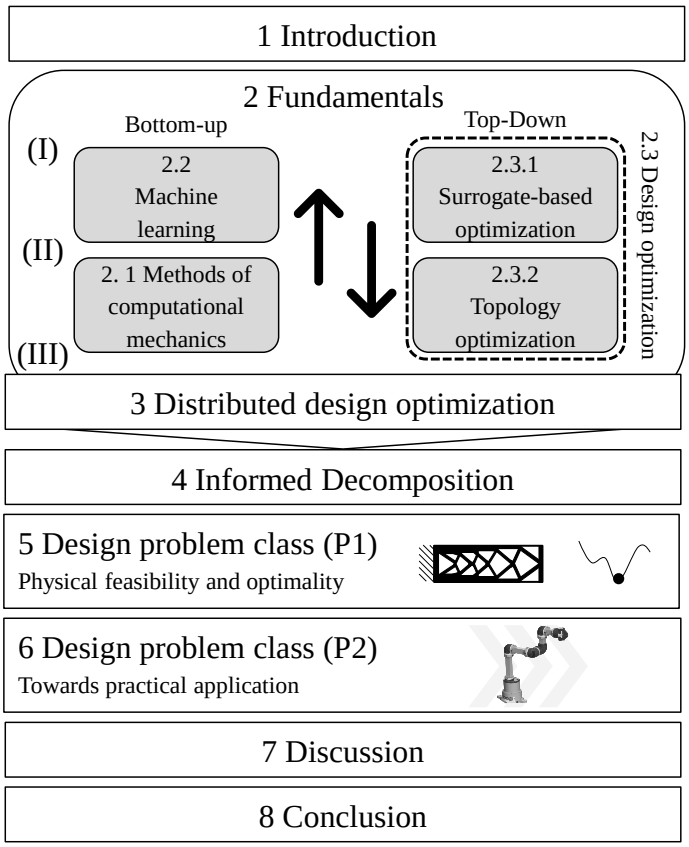


Figure 1.3: Structure of the thesis

2 Fundamentals

2.1 Methods of computational mechanics

2.1.1 Finite elements

The finite element method (FEM) is a numerical approach mainly used to solve physical problems in engineering analysis. Its basic principle is to discretize a continuum of a design domain Ω into small elements of known shape and behavior in order to approximate the system's response. Below, the relevant aspects of FEM for this work will be introduced. Further details and derivations can be found in, e.g., Hughes (2000), Zienkiewicz et al. (2005) or Bathe (2014).

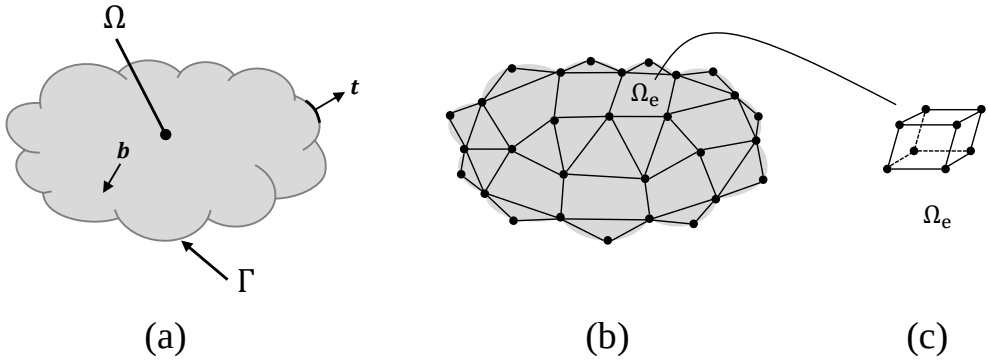


Figure 2.1: (a) Geometrical design domain Ω and boundary Γ with the applied body forces $\boldsymbol{b}$ and traction forces $\boldsymbol{t}$, (b) discretized design domain Ω, and (c) local element design domain Ω_e for a three-dimensional hexahedron finite element

For a general boundary value problem in linear elastostatics, Fig. 2.1 (a), the so-called weak form can be derived as

$$\int_\Omega \delta\boldsymbol{\varepsilon}^\top \boldsymbol{\sigma}\, d\Omega - \int_\Omega \delta\boldsymbol{d}^\top \boldsymbol{b}\, d\Omega - \int_\Gamma \delta\boldsymbol{d}^\top \boldsymbol{t}\, d\Gamma = 0, \tag{2.1}$$

where $\boldsymbol{\sigma}$, $\boldsymbol{\varepsilon}$, and $\boldsymbol{d}$ are the stresses, strains, and displacements at any point, $\boldsymbol{b}$ are the body forces and $\boldsymbol{t}$ the traction forces applied to the design domain Ω. $\delta\boldsymbol{\varepsilon}$ and $\delta\boldsymbol{d}$ can be chosen arbitrarily, and are usually referred to as the virtual strains and displacements, respectively. The term $\int_\Omega \delta\boldsymbol{\varepsilon}^\top\boldsymbol{\sigma}\, d\Omega$ represents the strain or internal energy, whereas the term $\int_\Omega \delta\boldsymbol{d}^\top\boldsymbol{b}\, d\Omega + \int_\Gamma \delta\boldsymbol{d}^\top\boldsymbol{t}\, d\Gamma$ determines the potential energy of the external loads.

The weak form for a given mechanical structure serves as a starting point for the FEM. The design domain Ω can be discretized into a finite number of smaller simpler elements, which are interconnected by nodes. An assembly of finite elements replaces the original geometry and is called a mesh, Fig. 2.1 (b). Next, an assumption about the elastic behavior of each element needs to be established, Fig. 2.1 (c). The specific element behavior can be described by local element stiffness matrices

$$\boldsymbol{K}_e = \int_{\Omega_e} \boldsymbol{B}^\top \boldsymbol{C}\boldsymbol{B}\, d\Omega_e, \tag{2.2}$$

where $\boldsymbol{B}$ is the element strain matrix and $\boldsymbol{C}$ is the material matrix of the constitutive equation. After having assigned the mechanical properties to the local elements, the global stiffness matrix $\boldsymbol{K}$ can be

assembled

$$\boldsymbol{K} = \bigwedge_{e=1}^{n_{\text{ele}}} \boldsymbol{K}_e. \tag{2.3}$$

The global stiffness matrix $\boldsymbol{K}$ approximates the elastic behavior of the design domain Ω based on the geometrical discretization. The respective discretized governing equations for the weak form are consequently

$$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}, \tag{2.4}$$

with $\boldsymbol{d}$ as the nodal displacement vector containing all displacements of the discretized design domain Ω and $\boldsymbol{f}$ as the corresponding load vector.

The stiffness matrix $\boldsymbol{K}$ possesses four important properties:

1. $\boldsymbol{K}$ must be symmetric, i.e., $\boldsymbol{K} = \boldsymbol{K}^\top$,
2. rigid body modes $\boldsymbol{\phi}_r$ result in zero forces, i.e., $\boldsymbol{K}\boldsymbol{\phi}_r = \boldsymbol{0}$,
3. $\boldsymbol{K}$ must be positive semi-definite, i.e., $\boldsymbol{d}^\top\boldsymbol{K}\boldsymbol{d} \geq 0$,
4. sparsity for assembled stiffness matrices $\boldsymbol{K} = \bigwedge_{e=1}^{n_{\text{ele}}} \boldsymbol{K}_e$.

Symmetry: This property follows directly from the Betti-Maxwell theorem of reciprocal work, which states that a displacement at location d_i caused by a unit load f_j at location j is equal to the displacement at location d_j caused by a unit load f_i at location i.

Rigid body modes: A structure without any defined support can move freely in space according to its rigid body modes $\boldsymbol{\phi}_r$. If dynamic effects are neglected, a displacement $\boldsymbol{d}$ based on the rigid body modes $\boldsymbol{\phi}_r$ does not require any force, hence $\boldsymbol{K}\boldsymbol{d}=\boldsymbol{0}$ holds, for $\boldsymbol{d}\neq\boldsymbol{0}$. From a mathematical point of view, an unsupported stiffness matrix $\boldsymbol{K}$ is therefore singular. Only by adding boundary conditions, such as a displacement boundary condition, this system can be solved.

Positive (semi-)definiteness: The positive (semi-)definiteness can be derived from physical considerations. Any displacement of a mechanical body which is not a rigid-body mode $\boldsymbol{\phi}_r$ must result in a strain energy $E=\frac{1}{2}\boldsymbol{d}^\top\boldsymbol{K}\boldsymbol{d}$. Since energy is by definition non-negative, $E\geq 0$, also $\frac{1}{2}\boldsymbol{d}^\top\boldsymbol{K}\boldsymbol{d} \geq 0$ must hold. This is the mathematical condition for positive semi-definiteness of matrices. From this property it can also be derived that a stiffness matrix $\boldsymbol{K}$ has only non-negative diagonal terms.

Sparsity: A matrix $\boldsymbol{K}$ consisting mainly of zero entries is called a sparse matrix. Sparsity allows for special storage and computation operations that are faster than a classical matrix operation. Due to the assembly process performed by $\bigwedge_{e=1}^{n_{\text{ele}}}$, the global stiffness matrix $\boldsymbol{K}$ is usually a sparse matrix. Only local degrees of freedom (not necessarily all) of the element stiffness matrix $\boldsymbol{K}_e$ are nonzero, as well as those connected to other elements by nodes.

2.1.2 Guyan reduction

For linear static problems, Guyan (1965) developed a master-slave elimination technique to reduce the computational cost of FEM computations. Sometimes the term Irons-Guyan reduction is also used since a similar method was developed simultaneously in Irons (1963) and Irons (1965). The main idea is to remove all degrees of freedom to which no loads or boundary conditions are applied. The nodes associated with degrees of freedom that are to be removed are called slave nodes, whereas the boundary conditions and loads are applied to the degrees of freedom of the master nodes, Fig. 2.2 (a).

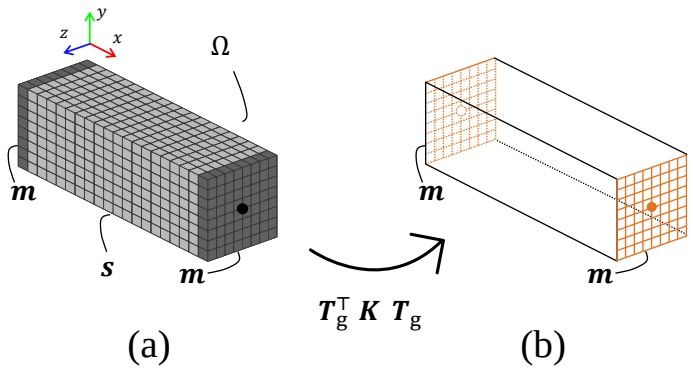


Figure 2.2: (a) Geometrical design domain Ω discretized by finite elements consisting of master and slave nodes, m and s, respectively, and (b) the reduced system $\boldsymbol{K}_g = \boldsymbol{T}_g^\top \boldsymbol{K}\, \boldsymbol{T}_g$ utilizing a Guyan reduction that removes the slave nodes s from Ω

The static problem of (2.4) can then be rewritten as

$$\begin{bmatrix} \boldsymbol{K}_{mm} & \boldsymbol{K}_{ms} \\ \boldsymbol{K}_{sm} & \boldsymbol{K}_{ss} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_m \\ \boldsymbol{d}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_m \\ \boldsymbol{0} \end{bmatrix}, \tag{2.5}$$

where the suffix m is used for the master degrees of freedom and the suffix s is used for the condensed slave degrees of freedom. Solving equation (2.5) in terms of $\boldsymbol{d}_m$ gives the following dependency for $\boldsymbol{d}_s$

$$\boldsymbol{d}_s = -\boldsymbol{K}_{ss}^{-1}\, \boldsymbol{K}_{sm}\, \boldsymbol{d}_m. \tag{2.6}$$

Introducing (2.6) into (2.5) leads to the reduced stiffness matrix $\boldsymbol{K}_\mathrm{g}$

$$\boldsymbol{K}_\mathrm{g} = \boldsymbol{K}_{mm} - \boldsymbol{K}_{ms}\, \boldsymbol{K}_{ss}^{-1}\, \boldsymbol{K}_{sm}. \tag{2.7}$$

Now, the reduced system

$$\boldsymbol{K}_\mathrm{g}\, \boldsymbol{d}_m = \boldsymbol{f}_m, \tag{2.8}$$

can be solved to determine the master nodes' displacements $\boldsymbol{d}_m$, Fig. 2.2 (b). Since all degrees of freedom contribute to the condensation process of $\boldsymbol{K}_\mathrm{g}$, there is no loss of accuracy (Guyan, 1965). The linear system of equations of (2.8) is therefore equivalent to the original problem (2.4).

The condensation process of the Guyan reduction can also be expressed with respect to a linear transformation $\boldsymbol{T}_\mathrm{g}$ (J.-G. Kim & Lee, 2014). The master displacements $\boldsymbol{d}_m$ are then related to the entire displacement vector $\boldsymbol{d}$ in the following way

$$\boldsymbol{d} = \quad \boldsymbol{T}_\mathrm{g}\, \boldsymbol{d}_m, \tag{2.9}$$

$$\boldsymbol{T}_\mathrm{g} = \begin{bmatrix} \boldsymbol{I} \\ -\boldsymbol{K}_{ss}^{-1}\, \boldsymbol{K}_{sm} \end{bmatrix}. \tag{2.10}$$

Using this transformation matrix $\boldsymbol{T}_\mathrm{g}$, the reduced stiffness matrix $\boldsymbol{K}_\mathrm{g}$ can be obtained in a more compact notation

$$\boldsymbol{K}_\mathrm{g} = \boldsymbol{T}_\mathrm{g}^\top\, \boldsymbol{K}\, \boldsymbol{T}_\mathrm{g}. \tag{2.11}$$

2.1.3 Rigid body elements

Rigid body elements are used in various engineering analysis tools to transfer loads or connect different components rigidly. Stemming from the FEM software Nastran, rigid body elements were introduced and applied to a variety of use cases. In the following, the concept of the so-called rigid body element 2 (RBE2), based on a master-slave elimination within a multi-point constraint, is explained according to Heirman & Desmet (2010) and G.-R. Liu & Quek (2013).

The RBE2 is a linear rigid element with one master node m and one or more slave nodes s. The degrees of freedom of the master node are often referred to as the independent degrees of freedom, whereas slave nodes possess dependent degrees of freedom. The rigidity between the master and slave nodes is prescribed by the following geometrical constraint on the slave displacements

$$\boldsymbol{d}_s = \begin{bmatrix} \boldsymbol{u}_s \\ \boldsymbol{\varphi}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_m + \boldsymbol{\varphi}_m \times [\boldsymbol{x}_s - \boldsymbol{x}_m] \\ \boldsymbol{\varphi}_m \end{bmatrix}, \tag{2.12}$$

where $\Delta\boldsymbol{x} = [\boldsymbol{x}_s - \boldsymbol{x}_m]$ represents the distance vector from the independent master to the dependent slave node, see Fig. 2.3 (a). The general displacement vector $\boldsymbol{d} = [\boldsymbol{u}, \boldsymbol{\varphi}]^\top$ consists of a translational $\boldsymbol{u}$ and rotational part $\boldsymbol{\varphi}$. Note that the above equation is only valid for small deformations.

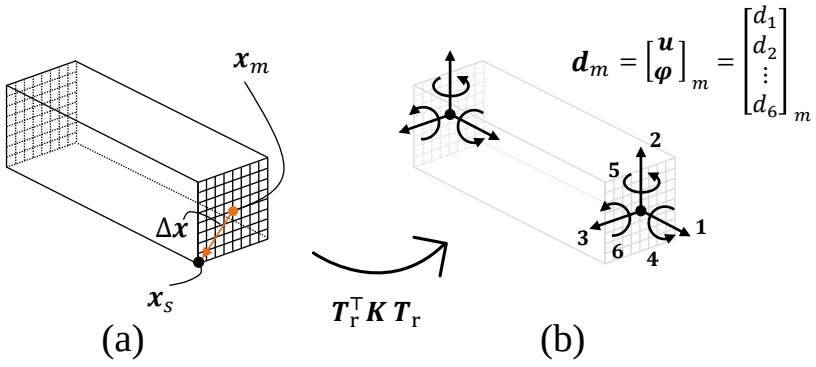


Figure 2.3: (a) Geometrical design domain Ω discretized by finite elements with one exemplary master node $\boldsymbol{x}_m$ (orange dot) and slave node $\boldsymbol{x}_s$ (black dot), defining the distance vector $\Delta\boldsymbol{x} = [\boldsymbol{x}_s - \boldsymbol{x}_m]$, and (b) the reduced system $\boldsymbol{K}_r = \boldsymbol{T}_r^\top \boldsymbol{K} \boldsymbol{T}_r$, which is computed using a multi-point constraint based on (2.12) for all slave nodes s belonging to the right and left sides of the design domain Ω

The geometrical constraint of (2.12) is incorporated into a homogeneous equality constraint

$$\boldsymbol{C} \quad \boldsymbol{d} = \boldsymbol{0}, \tag{2.13}$$

$$\begin{bmatrix} \boldsymbol{C}_m & \boldsymbol{C}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_m \\ \boldsymbol{d}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \end{bmatrix}, \tag{2.14}$$

where $\boldsymbol{C}$ is the constraint matrix and $\boldsymbol{d}$ is the general displacement vector, both ordered with respect to the master and slave nodes m and s, respectively. The constraint matrix $\boldsymbol{C}$ can be used to compute the new elastic system behavior $\boldsymbol{K}_\mathrm{r}$ based on a master-slave elimination

$$\boldsymbol{K}_\mathrm{r} = \boldsymbol{T}_\mathrm{r}^\top \boldsymbol{K} \boldsymbol{T}_\mathrm{r}, \tag{2.15}$$

where the condensation matrix $\boldsymbol{T}_\mathrm{r}$ and the total displacement $\boldsymbol{d}$ are similarly computed as in Section 2.1.2

$$\boldsymbol{d} = \quad \boldsymbol{T}_\mathrm{r}\boldsymbol{d}_m, \tag{2.16}$$

$$\boldsymbol{T}_\mathrm{r} = \begin{bmatrix} \boldsymbol{I} \\ -\boldsymbol{C}_s^{-1}\boldsymbol{C}_m \end{bmatrix}. \tag{2.17}$$

It should be noted that the introduction of a RBE2 formulation into a model artificially increases the stiffness of the structure by constraining some deformations that would be allowed in reality.

2.2 Machine learning

Machine learning models, or meta models, can be established to approximate the system behavior by fast but simple low-fidelity models. Within this work, the connection between level (II) and (I) of the given design problem is to be established by meta models.

The general idea of machine learning is to predict a specific unknown output z for given input data $\boldsymbol{y}$

$$z \approx \hat{z} = \hat{f}(\boldsymbol{y}), \tag{2.18}$$

based on previously provided data

$$\boldsymbol{Y} = [\boldsymbol{y}_1^\top, ..., \boldsymbol{y}_N^\top]^\top, \quad \boldsymbol{Y} \in \mathbb{R}^{N \times n_y}, \tag{2.19}$$
$$\boldsymbol{Z} = [z_1^\top, ..., z_N^\top]^\top, \quad \boldsymbol{Z} \in \mathbb{R}^{N \times n_z}. \tag{2.20}$$

The precise form of the meta model $\hat{f}(\boldsymbol{y})$ is determined during the training phase based on the training data $[\boldsymbol{Y}_{\text{tr}}, \boldsymbol{Z}_{\text{tr}}]$, which is a subset of the provided data of (2.19) and (2.20). Machine learning approaches can be divided into two different learning types: Supervised learning and unsupervised learning. Supervised learning deals with problems where a set of input $\boldsymbol{Y}$ and output data $\boldsymbol{Z}$ is known. This means that a priori knowledge of the relationship between both already exists. In unsupervised learning, the input data is available but there is no output data, hence $\boldsymbol{Z} = [\ ]$. Here, only relations between the input variables and the general structure of the data can be used to predict the output z. Depending on the form of the output z, one distinguishes between regression and classification problems in machine learning. In regression problems, the desired output z is in a continuous range, while classification uses a discrete assignment to the output variables, e.g., binary classification with two classes $z \in [-1, 1]$.

Once the meta model $\hat{f}(\boldsymbol{y})$ is trained, it can be tested by predicting the output z of new data from a test set $[\boldsymbol{Y}_{\text{te}}, \boldsymbol{Z}_{\text{te}}]$. It should be noted that testing should only be done with unseen data, i.e., the model should not be tested with data that has already been used for training. The ability to correctly predict new unseen data is referred to as generalization. In practical applications, the variability of the input vectors is so large that the training data may include only a tiny fraction of all possible input data, making generalization a central goal of machine learning. To ensure good generalization, it is important to obtain a sufficient amount of information about the problem through the training data. This is ensured by sampling methods, which provide the entire sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ and therefore play an important role in machine learning.

In general, there are a variety of different machine learning methods for both classification and regression problems, the interested reader is referred to Bishop (2006), Hastie et al. (2009) or James (2013). For this work, first, supervised learning for regression with artificial neural networks and binary classification with support vector machines are introduced, and afterwards different sampling strategies are explained that can provide the meta models with the needed sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$.

2.2.1 Regression with artificial neural networks

In the early years of artificial neural networks (1943-1958), several researchers were recognized for their pioneering contributions in this upcoming field (Haykin, 2009). Starting from the first idea of artificial neural networks as computing machines in McCulloch & Pitts (1943) to the derivation of the first hypothesis for self-organized learning in Hebb (1949) to the establishment of the first perceptron, i.e., the first artificial neural network by Rosenblatt (1958). However, it was not until the mid-1980s

that artificial neural networks gained attention by Rumelhart et al. (1986). In general, artificial neural networks can be used for both classification and regression problems. In the following, feed forward artificial neural networks for regression are presented based on Bishop (2006).

In linear regression models, a linear combination of n_b predefined nonlinear basis functions $\phi_i(\boldsymbol{y})$ and weights w_i are utilized to predict data

$$\hat{f}(\boldsymbol{y}, \boldsymbol{w}) = \sum_{i=1}^{n_\mathrm{b}} w_i \phi_i(\boldsymbol{y}). \tag{2.21}$$

However, these linear regression models are limited by the curse of dimensionality. To enable application to large-scale problems, it is necessary to adapt the basis functions to the training data. Therefore, basis functions $\phi_i(\boldsymbol{y})$ were extended to depend on additional parameters besides the weights w_i to improve their capacities.

In the field of artificial neural networks, the basic neural network is determined by a series of functional transformations. First, n_b so-called activations a_i are determined

$$a_i = \sum_{j=1}^{n_\mathrm{y}} w_{ij}^{(1)} y_j + w_{i0}^{(1)}, \tag{2.22}$$

where w_{ij} is referred to as weights, w_{i0} as biases, and the superscript (1) corresponds to the first layer of the network.

Next, for each activation a_i a differentiable, nonlinear activation function $h(\,\cdot\,)$ is utilized

$$b_i = h(a_i). \tag{2.23}$$

Each output b_i can be seen as a basis function $\phi_i(\boldsymbol{y})$ of equation (2.21) and are called hidden units in the field of neural networks. A classical choice of activation function is the logistic sigmoid or the tangens hyperbolicus function.

In accordance with the idea of linear regression, those intermediate outputs b_i are linearly combined to the overall output $\hat{z}$

$$\hat{z} = \sum_{i=1}^{n_\mathrm{b}} w_i^{(2)} b_i + w_0^{(2)}. \tag{2.24}$$

Note that for regression problems usually only one scalar output $\hat{z}$ is chosen, in contrast to multiple ones within classification problems. In total, the overall network function $\hat{f}(\boldsymbol{y}, \boldsymbol{w}) = \hat{z}$ is then

$$\hat{f}(\boldsymbol{y}, \boldsymbol{w}) = \sum_{i=1}^{n_\mathrm{b}} w_i^{(2)}\, h\left(\sum_{j=1}^{n_\mathrm{y}} w_{ij}^{(1)} y_j + w_{i0}^{(1)} \right) + w_0^{(2)}, \tag{2.25}$$

see also Fig. 2.4, or simplified

$$\hat{f}(\boldsymbol{y}, \boldsymbol{w}) = \sum_{i=0}^{n_\mathrm{b}} w_i^{(2)}\, h\left(\sum_{j=0}^{n_\mathrm{y}} w_{ij}^{(1)} y_j \right), \tag{2.26}$$

by introducing additional parameters $y_0 = b_0 = 1$.

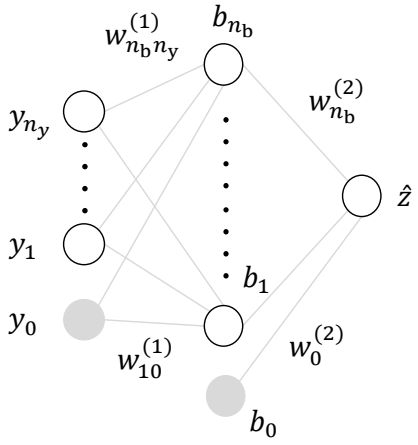


Figure 2.4: Network diagram for the two-layer neural network corresponding to equation (2.25). The input y_j, hidden units b_i, and output variables $\hat{z}$ are represented by nodes, and the weight parameters w_{ij} and w_i are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables y_0 and b_0

Equation (2.25) and (2.26) are nonlinear functions from a set of input variables $\boldsymbol{y}$ to a total output variable $\hat{z}$ given by a vector of adjustable parameters $\boldsymbol{w}$. The term feedforward refers to the fact that the architecture has no loops, i.e., the outputs are deterministic functions of the inputs. Considering the activation functions of all hidden units of a network to be linear, one can always find an equivalent network without hidden units for such a network. For one layer, Minsky & Papert (1969) have shown the limitations of neural networks in learning the relationship between inputs and outputs by demonstrating that neural networks are unable to classify patterns of nonlinearly separable classes. However, with multiple layers, artificial neural networks are universal approximators that can successfully approximate any function if the parameters are chosen appropriately (Hornik et al., 1989).

The network architecture of Fig. 2.4 is the most commonly used architecture with input y_j, a so-called hidden layer with hidden units b_i, and an output $\hat{z}$. However, it can be easily extended, e.g., by considering additional hidden layers, each consisting of a weighted linear combination followed by an element-wise transformation with a nonlinear activation function $h(\cdot)$. Note that there are different terminologies in the literature with respect to counting the number of layers in such networks. We adhere to the terminology of Bishop (2006), in which the architecture of Fig. 2.4 is a two-layer network because the number of layers of adaptive weights $\boldsymbol{w}$ is important for determining the network properties.

To determine the weights $\boldsymbol{w}$ of an artificial neural network, an optimization objective needs to be defined. For regression problems with a given training data $[\boldsymbol{y}, z]_A$, for A=1, ..., N, usually the residual sum of squares (RSE) is minimized

$$\min_{\boldsymbol{w}} \mathrm{RSE} = \sum_{A=1}^{N} \left(\hat{f}(\boldsymbol{y}_A, \boldsymbol{w}) - z_A \right)^2, \tag{2.27}$$

where z_A are the observed target values and $\hat{f}(\boldsymbol{y}_A, \boldsymbol{w})$ are the predictions of a trained neural network. Note that the nonlinearities of $\hat{f}(\boldsymbol{y}_A, \boldsymbol{w})$ lead to a nonconvex optimization problem with respect to the error RSE.

The weights $\boldsymbol{w}$ are determined by incorporating first and sometimes second order information into the optimization. There are two types of optimization approaches for training artificial neural networks: batch methods and online methods. Batch methods use the entire dataset at once, which has the

advantage of accurate gradient estimation and parallelization but requires a large amount of memory. Due to non-convexity, multiple starting points are used in practice to at least have a comparison between different local minima to ensure a good enough solution. In contrast, online methods use only one data point at a time and are therefore particularly suitable for large datasets. They are easy to implement and deal more efficiently with redundancies in the training data. In addition, there is the possibility of escaping local minima (Bishop, 2006; Haykin, 2009). In order to evaluate the derivatives of the objective function efficiently, the so-called error backpropagation algorithm can be applied. Rumelhart et al. (1986) proposed the first backpropagation algorithm involving a gradient-descent method. If second-order information is necessary, the Hessian matrix $\boldsymbol{H}$ can be computed using backpropagation, too. Because the computation of the Hessian matrix is expensive, approximation schemes are utilized, as e.g., the Levenberg–Marquardt approximation.

Instead of the RSE, the mean squared error (MSE) can also be used, which represents the average of the squared errors

$$\mathrm{MSE} = \frac{1}{N} \sum_{A=1}^{N} \left(\hat{f}(\boldsymbol{y}_A, \boldsymbol{w}) - z_A \right)^2 . \tag{2.28}$$

For more details on how to train an artificial neural network by means of backpropagation, the interested reader is referred to Bishop (2006). Besides the RSE and MSE, other performance measurements can be utilized after the training process to evaluated the resulting artificial neural network. For instance, the coefficient of determination, denoted as R^2, is often used to assess the accuracy of a regression model

$$R^2 = 1 - \frac{\mathrm{RSE}}{\mathrm{TSS}}, \tag{2.29}$$

where $\mathrm{TSS} = \sum_{A=1}^{N} (\hat{f}(\boldsymbol{y}_A, \boldsymbol{w}) - \bar{z})^2$ is the total sum of squares. R^2 takes only values between 0 and 1 and measures the proportion of variability in $\boldsymbol{z}$ that can be explained using $\boldsymbol{y}$ (James, 2013).

While the number of input and output units in a neural network is generally determined by the dimensionality of the problem and the weights $\boldsymbol{w}$ by solving the optimization problem (2.27), the number of hidden layers and units can be adjusted independently to achieve the best predictive performance. These so-called hyperparameters, whose values are set before the training process, have a significant influence on the results (Bishop, 2006). Various techniques for the determination of hyperparameters exist, a simple way is carrying out a full-factorial design with all hyperparameters and choosing the configuration with the best performance with respect to the specified performance values.

2.2.2 Binary classification with support vector machines

The support vector machine is a supervised machine-learning method initially developed for binary classification problems over a duration of 30 years from 1965-1995 (Vapnik & Kotz, 2006). It is based on three major developments:

1. an algorithm for optimal separating hyperplanes for linear classification problems (Vapnik & Chervonenkis, 1974) using the Vapnik–Chervonenkis theory,
2. the extension to nonlinear classification problems by constructing a hyperplane using the kernel trick in Boser et al. (1992), and
3. the generalization of the maximal margin idea for non-separable and nonlinear classification problems in Corinna Cortes & Vladimir Vapnik (1995), i.e., the actual support vector machine (SVM).

SVMs can in general be applied to both, regression and classification tasks. In the following, a SVM for binary classification tasks is reviewed.

1. Linear classification problems

A binary classification problem ($z \in [-1, 1]$) is linearly separable, if for a given training data $[\boldsymbol{y}, z]_A$ a scalar b and a vector $\boldsymbol{w}$ exist, for which

$$\begin{aligned} \boldsymbol{w}\boldsymbol{y}_A^\top + b \geq \quad 1, \quad \text{if } z_A = 1, \\ \boldsymbol{w}\boldsymbol{y}_A^\top + b \leq -1, \quad \text{if } z_A = -1, \end{aligned} \tag{2.30}$$

holds, see Fig. 2.5 (b).

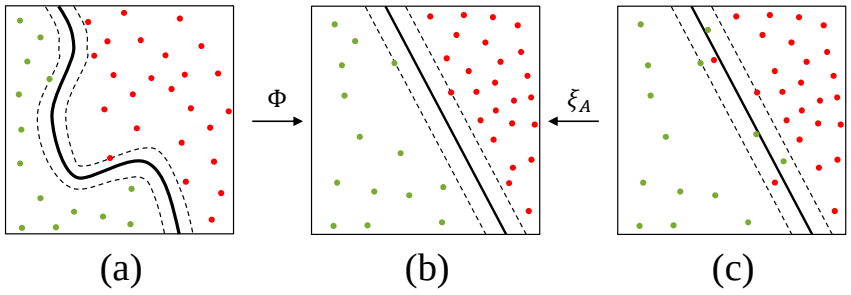


Figure 2.5: Support vector machine for (a) nonlinear classification problems using the transformation $\boldsymbol{\Phi}(\boldsymbol{y})$ to convert it to (b) a linear classification problem with perfectly separable training data. (c) Non-separable training data can be processed using slack variables ξ_A to shift data to the hard margin boundary

The unique solution that separates the training data with a maximal margin M perfectly is called hard-margin hyperplane

$$\boldsymbol{w}^* \boldsymbol{y}^\top + b^* = 0, \tag{2.31}$$

where the margin M is defined as $M(\boldsymbol{w}, b) = 2/|\boldsymbol{w}|$.

In order to maximize this margin M, $\boldsymbol{w}$, and b have to be determined such that $|\boldsymbol{w}| = \boldsymbol{w}\boldsymbol{w}^\top$ is minimized, while ensuring a clear separation between the two classes. The underlying optimization problem is therefore a quadratic programming problem of the following form

$$\begin{aligned} &\min_{\boldsymbol{w},b} \tfrac{1}{2}\boldsymbol{w}\boldsymbol{w}^\top, \\ \text{s. t.:} \quad &z_A \left(\boldsymbol{w}\boldsymbol{y}_A^\top + b\right) \geq 1, \quad \text{for } A = 1, ..., N, \end{aligned} \tag{2.32}$$

where

$$z_A \left(\boldsymbol{w}\boldsymbol{y}_A^\top + b\right) \geq 1, \tag{2.33}$$

represents the condition for linear separation of (2.30). The vectors $\boldsymbol{y}_A$ closest to the hyperplane, i.e., where the inequality constraint is active $z_A \left(\boldsymbol{w}\boldsymbol{y}_A^\top + b\right) = 1$, are called support vectors. The solution of this quadratic optimization problem can be written as a linear combination of the support vectors $\boldsymbol{y}_A$, which determines the hard-margin hyperplane

$$\boldsymbol{w} = \sum_{A=1}^{N} z_A \lambda_A \boldsymbol{y}_A. \tag{2.34}$$

2. Nonlinear classification problems

The method so far, only applies for linear classification problems. In Boser et al. (1992), an efficient method for computing hyperplanes for nonlinear classification problems in a linear feature space was introduced. Here, the input space is transformed using a nonlinear predefined function $\boldsymbol{\Phi}(\boldsymbol{y})$, see Fig. 2.5 (a-b),

$$\boldsymbol{w}\,\boldsymbol{\Phi}^{\top}(\boldsymbol{y}) + b = 0. \tag{2.35}$$

The respective optimization problem to solve the nonlinear classification problem is

$$\begin{aligned} &\min_{\boldsymbol{w},b} \tfrac{1}{2}\boldsymbol{w}\boldsymbol{w}^{\top}, \\ &\text{s. t.:} \quad z_A\,(\boldsymbol{w}\,\boldsymbol{\Phi}^{\top}(\boldsymbol{y}_A) + b) \geq 1, \quad \text{for } A = 1, ..., N. \end{aligned} \tag{2.36}$$

The solution vector $\boldsymbol{w}$ can then be rewritten as

$$\boldsymbol{w} = \sum_{A=1}^{N} z_A \lambda_A \boldsymbol{\Phi}(\boldsymbol{y}_A). \tag{2.37}$$

3. Non-separable and nonlinear classification problems

If the classification data cannot be separated perfectly, an error margin needs to be introduced, see Fig 2.5 (c). Corinna Cortes & Vladimir Vapnik (1995) developed the general procedure for nonlinear, non-separable training data, called SVM. The inequality constraint of (2.36) is therefore extended with slack variables ξ_A

$$z_A\,(\boldsymbol{w}\,\boldsymbol{\Phi}^{\top}(\boldsymbol{y}_A) + b) \geq 1 - \xi_A, \tag{2.38}$$

which allows for classification error during the training process, i.e., that data points $\boldsymbol{y}_A$ are by ξ_A on the wrong side of the hyperplane. The hyperplane is therefore called soft-margin hyperplane. Instead of only maximizing the minimal distance, one tries to solve the classification problem with a minimal number of errors

$$\Theta(\boldsymbol{\xi}) = \sum_{A=1}^{N} \xi_A. \tag{2.39}$$

The general SVM optimization problem to determine the optimal soft-margin hyperplane then reads

$$\begin{aligned} &\min_{\boldsymbol{w},b} \tfrac{1}{2}\boldsymbol{w}\boldsymbol{w}^{\top} + C\,\Theta(\boldsymbol{\xi}), \\ &\text{s. t.:} \quad z_A\,(\boldsymbol{w}\,\boldsymbol{\Phi}^{\top}(\boldsymbol{y}_A) + b) \geq 1 - \xi_A, \quad \text{for } A = 1, ..., N, \\ &\qquad\qquad \xi_A \geq 0, \end{aligned} \tag{2.40}$$

where C is a constant penalty factor for slack variables ξ_A. The larger the penalty factor C, the larger the penalty for data points $\boldsymbol{y}_A$ that are misclassified during optimization. Eventually, fewer data points will cross the hyperplane boundary. However, this comes at the cost of making the SVM optimization problem to be solved more complex, i.e., more nonlinear. The smaller C is, the more data points will cross the boundary, but the final model will be smoother. For imbalanced training data, meaning that a dataset has an uneven distribution between its classes, the penalty constant C can be also used to improve the classification performance of the SVM.

In contrast to the optimization problem (2.27) of neural networks from Section 2.2.1, the SVM optimization problem (2.40) is convex, thus simplifying the training process. The solution vector of (2.37) defines now the soft-margin hyperplane, which may separate some training data incorrectly. The resulting SVM is then

$$\hat{z} = \hat{f}(\boldsymbol{\Phi}(\boldsymbol{y}), \boldsymbol{w}), \quad \text{for } \hat{z} \in [-1, 1]. \tag{2.41}$$

In order to assess the performance of the resulting SVM with respect to the sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$, different performance measure can be utilized (James, 2013). The most common performance measure is the accuracy ACC, which is the fraction of errors that are made when the estimates $\hat{z}_A = \hat{f}(\boldsymbol{\Phi}(\boldsymbol{y}_A), \boldsymbol{w})$ are compared to the training observations z_A

$$\text{ACC} = \frac{1}{N} \sum_{A=1}^{N} I \left(\hat{f}(\boldsymbol{\Phi}(\boldsymbol{y}_A), \boldsymbol{w}) - z_A \right). \tag{2.42}$$

Here $I(\hat{f}(\boldsymbol{\Phi}(\boldsymbol{y}_A), \boldsymbol{w}) - z_A)$ is an indicator variable that is I=1, if $z_A = \hat{z}$ and I=0, if $z_A \neq \hat{z}$.

Closely related to the accuracy ACC are the errors for correctly (true) or incorrectly (false) predicted classes in binary classification. The two classes of $z \in [-1, 1]$ are often referred to as negative and positive. The so-called confusion matrix, Table 2.1, contains the rates of the four different classification conditions, i.e., true positive (TP), true negative (TN), false positive (FP), and false negative (FN). N_p and N_n are the number of positive and negative samples, respectively, and $N = N_\text{p} + N_\text{n}$.

Table 2.1: Confusion matrix in binary classification

		Predicted class: Positive	Predicted class: Negative
Actual class: Positive		True positive rate $\text{TPR} = \frac{\text{TP}}{N_\text{p}}$	False negative rate $\text{FNR} = \frac{\text{FN}}{N_\text{p}}$
Actual class: Negative		False positive rate $\text{FPR} = \frac{\text{FP}}{N_\text{n}}$	True negative rate $\text{TNR} = \frac{\text{TN}}{N_\text{n}}$

While the optimization problem (2.40) determines the best SVM hyperplane for given data $[\boldsymbol{Y}, \boldsymbol{Z}]$, some hyperparameters must also be specified for the SVM. For example, the choice of the kernel function $\boldsymbol{\Phi}(\boldsymbol{y}, k_\text{s})$, which can also depend on other parameters k_s and the so-called box constraint, i.e., the previously introduced cost C. The hyperparameters are usually determined by cross-validation procedures, e.g., k-fold cross-validation, for more information see James (2013).

2.2.3 Sampling methods

In order to ensure a good generalization of the trained meta models $\hat{f}$ for a given training data $[\boldsymbol{Y}, \boldsymbol{Z}]$, a large number of different sampling techniques exist. Instead of sampling, often also the term design of experiment (DoE) is used in the field of machine learning. In the following four different sampling techniques are introduced:

1. Random sampling,
2. Latin hypercube sampling,
3. Random undersampling, and
4. Informed undersampling,

see also Fig. 2.6.

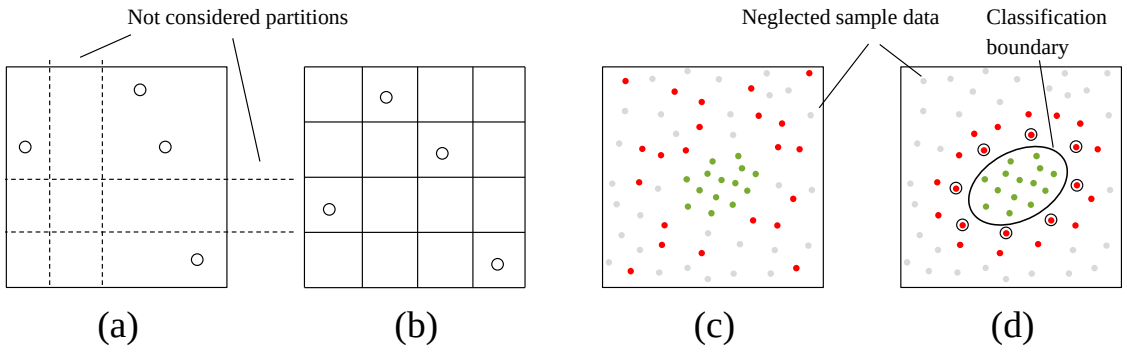


Figure 2.6: (a) Random sampling, (b) latin hypercube sampling for $N = 4$ partitions, (c) random undersampling, and (d) informed undersampling using SVM

1. Random sampling

Random sampling is the most commonly used sampling technique. Let the inputs $\boldsymbol{y}_A \in \mathbb{R}^{1\times n_y}$ be a random sample from a known probability distribution, then the sample input data, for $A = 1, ..., N$ in each dimension j, can be computed as

$$y_{j,A} = y_{\mathrm{lb},j} + \xi \left(y_{\mathrm{ub},j} - y_{\mathrm{lb},j}\right), \quad \text{for } 0 \leq \xi \leq 1, \tag{2.43}$$

where ξ is a random variable with the probability distribution $F(\xi)$, see Fig. 2.6 (a). Besides machine learning, random sampling is used in a variety of Monte Carlo methods (Andrieu et al., 2003).

2. Latin hypercube sampling

In McKay et al. (1979), a method called latin hypercube sampling was introduced and proved that the performance of meta models $\hat{f}$ can be improved, if certain monotonicity conditions hold. For a given sample number N, each dimension j of the design space of $\boldsymbol{y} \in \mathbb{R}^{1\times n_y}$ is first partitioned into N disjoint subspaces or partitions with equal probabilities $1/N$. Each input y_j is then sampled once for each partition $P = [1, ..., N]$

$$y_{j,P} = y_{\mathrm{lb},j} + \frac{(P-\xi)}{N}\left(y_{\mathrm{ub},j} - y_{\mathrm{lb},j}\right), \qquad \text{for } 0 \leq \xi \leq 1. \tag{2.44}$$

Each component $y_{j,P}$ for each partition P is then randomly combined to an input sample vector $\boldsymbol{y}_A$

$$\boldsymbol{y}_A = [y_{1,P_\xi}, ..., y_{n_y,P_\xi}], \tag{2.45}$$

where P_ξ takes random values of $P \in [1, ..., N]$. In contrast to random sampling, Fig. 2.6 (a), using latin hypercube sampling ensures that the complete sample input data $\boldsymbol{Y}$ consist of components $y_{j,P}$ of each partition P, see also Fig. 2.6 (b).

Classification: Learning from imbalanced data

Most machine learning algorithms for classification assume that the number of sample points in each class is approximately equal. However, training data is often imbalanced, meaning that a dataset has an uneven distribution between its classes. A fundamental problem with imbalanced training data is that it significantly affects the performance of most standard learning algorithms. According to Krawczyk (2016), three different techniques are commonly used to address this impact: 1. data-level methods that modify the sample set to balance distributions and/or remove samples, 2. algorithm-level methods

that directly modify existing learning algorithms to reduce bias against majority classes and adapt them to evaluate data with skewed distributions, e.g., the cost C of SVMs of (2.40), and 3. hybrid methods that combine the benefits of the previous two groups. At the data-level, various sampling techniques attempt to apply certain heuristics to directly provide a well-balanced dataset to the machine learning algorithm (He & Garcia, 2009; Krawczyk, 2016). Two data-level methods are presented below.

3. Random undersampling

A natural way to deal with this problem is undersampling. Undersampling attempts to compensate for the imbalance by randomly removing data from the majority class of the original dataset. Undersampling, however, has a relatively obvious problem: by removing sample data from the majority class, the classifier may miss important information (He & Garcia, 2009). The general concept of undersampling is illustrated in Fig. 2.6 (c).

4. Informed undersampling

To overcome this drawback, informed undersampling can be used. The goal of informed undersampling is to avoid or minimize information loss by including additional information in the sampling process. One possibility of informed undersampling are active-learning strategies. Active-learning refers to algorithms that automatically select data points from which to learn. A special class of active-learning strategies is called uncertainty sampling. In uncertainty sampling, the algorithm retains only the sample points where the classifier is most uncertain and neglects all other sample points. This is done by evaluating the sample points $\boldsymbol{y}_A$ by a meta model $\hat{f}$ and selecting those closest to the classification boundary approximated by the meta model $\hat{f}$, see also Fig. 2.6 (d). SVMs are particularly well suited for this type of informed undersampling because the mathematical definition of the separating hyperplane makes the distance calculation trivial (Ertekin et al., 2007; Kremer et al., 2014).

For more information on different sampling techniques or also other ways of dealing with imbalanced training datasets as algorithm-level and hybrid methods, please see He & Garcia (2009) or Krawczyk (2016).

2.3 Design optimization

Design optimization is a mathematical design method to seek the optimal or best design within the available means (Papalambros & Wilde, 2018; Martins & Ning, 2022). The mathematical standard formulation for optimization is typically stated as a minimization problem in the negative null form

$$\begin{aligned} & \min_{x_j} f(x_j), \\ \text{s. t.:} \quad & g_k(x_j) \leq 0, && \text{for } k = 1, ..., n_\text{g}, \\ & h_l(x_j) = 0, && \text{for } l = 1, ..., n_\text{h}, \\ & x_{\text{lb},j} \leq x_j \leq x_{\text{ub},j}, && \text{for } j = 1, ..., n_\text{x}. \end{aligned} \tag{2.46}$$

To distinguish and modify designs, the design variables $\boldsymbol{x} = [x_1, ..., x_{n_\text{x}}]$ are introduced that describe the design of the given problem. The best design is hereby evaluated with the objective function $f(\boldsymbol{x})$ that depends on these design variables and is formulated by the design engineer to realize a given objective. The term available means refers to the constraints $\boldsymbol{g}(\boldsymbol{x})=[g_1, ..., g_{n_\text{g}}] \leq \boldsymbol{0}$ and $\boldsymbol{h}(\boldsymbol{x})=[h_1, ..., h_{n_\text{h}}]=\boldsymbol{0}$ and the bounds $[\boldsymbol{x}_\text{lb}, \boldsymbol{x}_\text{ub}]$ posed on the design problem. These are typically problem specific restriction that cannot be changed by the designer. The term negative null form stems from the fact that the inequality constraints are formulated as $\boldsymbol{g}(\boldsymbol{x}) \leq 0$. We can now, for example, recognize the monolithic optimization problem (1.3) of Section 1.2 as a minimization problem in the negative null form.

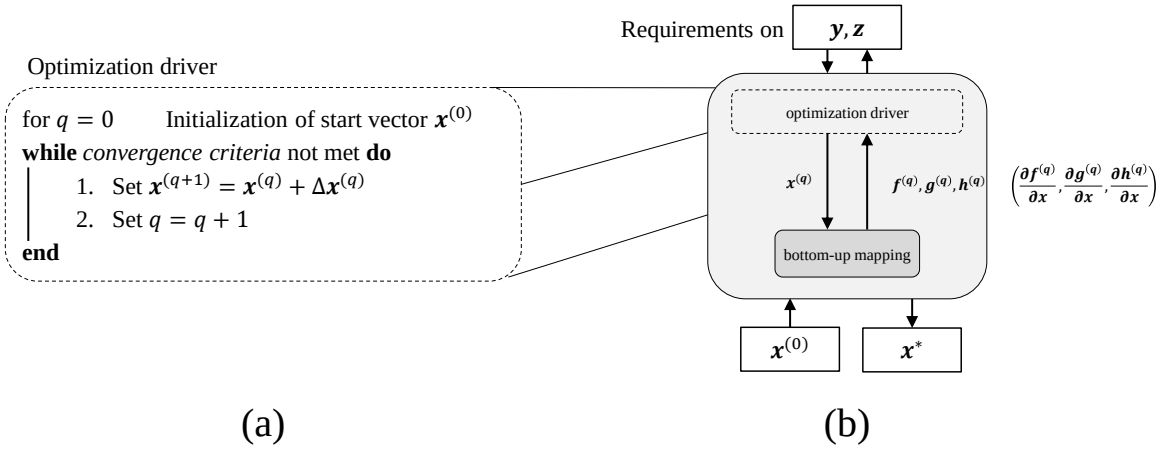


Figure 2.7: (a) Fundamental steps of a general optimization algorithm carried out by the optimization driver and (b) flowchart of an optimization algorithm architecture with the two main parts: optimization driver and bottom-up mapping

After having set up an optimization problem (2.46), the actual task of finding the optimal design

$$\boldsymbol{x}^*, \tag{2.47}$$

needs to be solved. In practice, the increasing complexity of design optimization problems makes an analytical solution procedure impractical. Therefore, numerical iterative optimization algorithms are applied to solve the problem at hand. The fundamental steps of an optimization algorithm can be seen in Fig 2.7 (a).

For q=0, the optimization driver initializes a start vector

$$\boldsymbol{x}^{(0)}, \tag{2.48}$$

which is then changed in each new iteration q according to algorithm-specific rules $\Delta \boldsymbol{x}^{(q)}$

$$\boldsymbol{x}^{(q+1)} = \boldsymbol{x}^{(q)} + \Delta \boldsymbol{x}^{(q)}, \tag{2.49}$$

$$q = q + 1, \tag{2.50}$$

until a predefined convergence criteria is met. Note that depending on the mathematical nature of the design problem, the choice of the start vector $\boldsymbol{x}^{(0)}$ as well as the utilized optimization algorithm can have a significant impact on the final optimization result $\boldsymbol{x}^*$.

The general flowchart of an optimization algorithm is shown in Fig 2.7 (b). The two main parts of an optimization algorithm are

1. optimization driver and
2. bottom-up mapping.

The previously mentioned optimization driver steers the optimization based on the given top-level requirements for $\boldsymbol{y}$ and $\boldsymbol{z}$ from the top down to the optimal bottom-level solution $\boldsymbol{x}^*$, which is why the term top-down is also sometimes used here. In contrast, the corresponding bottom-up mappings compute the actual top-level response values $\boldsymbol{y}$ and $\boldsymbol{z}$ based on the chosen bottom-level design variables $\boldsymbol{x}$, hence the term bottom-up. In engineering, these bottom-up mappings are mostly high-dimensional computer models, as FEM models of Section 2.1, that must be run multiple times during the course of the algorithm, hence limiting the usage of design optimization due to too high computational cost. If the computational cost of high-fidelity models is too large, often a so-called surrogate-based optimization is utilized. In surrogate-based optimization, low-fidelity meta models are established to approximate the system behavior by faster but simpler low fidelity models, as introduced in Section 2.2. Note that bottom-up mappings are also often just called analyses in the optimization community.

Many different classifications of optimization algorithms exist (Papalambros & Wilde, 2018; Martins & Ning, 2022). Two important ways of classification are:

1. gradient-based vs. gradient-free algorithms,
2. local vs. global search algorithms.

Gradient-based algorithms rely on local gradient ($\partial / \partial x$) or curvature ($\partial^2 / \partial x^2$) information for changes $\Delta \boldsymbol{x}^{(q)}$ in the current design $\boldsymbol{x}^{(q)}$. In general, gradient-based algorithms scale better to problems with many design variables than gradient-free algorithms. However, since first or second order information is required, this information must be readily available when using such a method. Unfortunately, often black-box functions with a simple input-output structure are used, hence lacking analytical derivatives. Here, the gradients must be numerically approximated by finite difference schemes, which is computationally expensive. Moreover, gradient-based algorithms tend to converge to the nearest local minimum rather than the global minimum. Another problem arises when the objective function is non-smooth, i.e., discontinuities exist, or the values of the design variables take only integer values, leading to a discrete problem. In all these cases, gradient-free algorithms are preferred. They rely only on zero-order information of the used bottom-up mappings $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$, and $\boldsymbol{h}(\boldsymbol{x})$. By simultaneously evaluating many different designs $\boldsymbol{x}_A$, it is attempted to identify the optimal global design $\boldsymbol{x}^*_{\text{glob}}$.

The terms local and global search algorithms refer to the way of how the design space is searched. Local search algorithms start from one single start vector $\boldsymbol{x}^{(0)}$ and take a series of steps q that are supposed to converge to the closest local optimum $\boldsymbol{x}^*_{\text{loc}}$. In contrast, global search algorithms try to scan the whole design space utilizing multiple designs $\boldsymbol{x}_A$ in order to find the global optimum $\boldsymbol{x}^*_{\text{glob}}$, however global convergence can also not be ensured. Since gradient-based algorithms rely typically on local information and gradient-free algorithms indeed scan a broader area of the design space, the terms local and gradient-based, and global and gradient-free are often used synonymously. Yet, sometimes gradient-based algorithms use global search information and gradient-free algorithm rely on local-search strategies (Martins & Ning, 2022).

2.3.1 Surrogate-based optimization

The system level optimization problem we want to explore is based on meta models. In this context, the term surrogate-based optimization is often used in engineering. The goal is to create meta models that are much faster to compute than the original models, but still maintain sufficient accuracy (Martins & Ning, 2022). This does not necessarily mean that the responses of the meta models are approximations; they can still be 100% accurate. The optimization problem (2.46) therefore only changes with respect to the used bottom-up mappings $\hat{f}$, $\hat{g}$, and $\hat{h}$

$$\begin{aligned} & \min_{y_j} \hat{f}(y_j), \\ \text{s. t.:} \quad & \hat{g}_k(y_j) \leq 0, && \text{for } k = 1, ..., n_\text{g}, \\ & \hat{h}_l(y_j) = 0, && \text{for } l = 1, ..., n_\text{h}, \\ & y_{\text{lb},j} \leq y_j \leq y_{\text{ub},j}, && \text{for } j = 1, ..., n_\text{y}. \end{aligned} \tag{2.51}$$

Since the surrogate-based system optimization problem is to be carried out between the hierarchy level (I) and (II) of the design problem, the design variables for the system level optimization are now the component performances $\boldsymbol{y}$ of the hierarchy level (II), see Section 1.2. Instead of the original high-fidelity bottom-up mappings, meta models are used and the objective and constraint functions are now $\hat{f}(\boldsymbol{y})$, $\hat{g}(\boldsymbol{y})$, and $\hat{h}(\boldsymbol{y})$.

To avoid that an optimal design $\boldsymbol{x}^*$ is not already ruled out at the system level, we want to ensure global convergence $\boldsymbol{y}^*$ for our given system optimization design problem. Therefore, a gradient-free global search algorithm called particle swarm optimization (PSO) is used. Since the number of design variables n_y of hierarchy level (II) is relatively small and the computational cost of the used meta models $\hat{f}(\boldsymbol{y})$, $\hat{g}(\boldsymbol{y})$, and $\hat{h}(\boldsymbol{y})$ should also be comparatively small, evaluating multiple designs $\boldsymbol{y}_A$ during the PSO does not pose any problem in terms of computational cost.

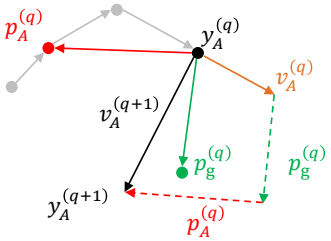


Figure 2.8: Geometrical representation of the update scheme of PSO to compute the new position $\boldsymbol{y}_A^{(q+1)}$ based on the current position $\boldsymbol{y}_A^{(q)}$, the velocity $\boldsymbol{v}_A^{(q)}$, the particle's best position $\boldsymbol{p}_A^{(q)}$, and the swarm's best position $\boldsymbol{p}_\text{g}^{(q)}$

Even though the PSO is a gradient-free and global-search algorithm, it still incorporates local-search information. It was initially proposed by Kennedy & Eberhart (1995) and further developed in Eberhart & Kennedy (1995) and Shi & Eberhart (1998). It is inspired by the idea of simulating social behavior in nature. In contrast to classical gradient-based optimization algorithms, PSO starts with an initial set of start vectors $\boldsymbol{Y}^{(0)} = [\boldsymbol{y}_1^\top, ..., \boldsymbol{y}_{N_\text{s}}^\top]^\top$. One design variable vector $\boldsymbol{y}_A$, for A=1, ..., N_s, is referred to as a particle, whereas the set of particles in one iteration $\boldsymbol{Y}^{(q)}$ is called a swarm in the context of PSO. During the course of the algorithm, each particle $\boldsymbol{y}_A^{(q)}$ takes individual steps based on the following update scheme

$$\boldsymbol{v}_A^{(q+1)} = w\,\boldsymbol{v}_A^{(q)} + \xi_1\,(\boldsymbol{p}_A^{(q)} - \boldsymbol{y}_A^{(q)}) + \xi_2\,(\boldsymbol{p}_\text{g}^{(q)} - \boldsymbol{y}_A^{(q)}), \tag{2.52}$$

$$\boldsymbol{y}_A^{(q+1)} = \boldsymbol{y}_A^{(q)} + \boldsymbol{v}_A^{(q+1)}, \tag{2.53}$$

where $\boldsymbol{y}_A^{(q)}$ is the current position, $\boldsymbol{v}_A^{(q)}$ the velocity, $\boldsymbol{p}_A^{(q)}$ the particle's best position and $\boldsymbol{p}_{\mathrm{g}}^{(q)}$ the swarm's best position, which are used to calculate the new velocity $\boldsymbol{v}_A^{(q+1)}$ and thus the new position $\boldsymbol{y}_A^{(q+1)}$, see also Fig. 2.8.

The three components of the velocity update can be interpreted as:

1. Inertia: $\boldsymbol{v}_A^{(q)}$,
2. Cognition: $\boldsymbol{p}_A^{(q)} - \boldsymbol{y}_A^{(q)}$,
3. Social: $\boldsymbol{p}_{\mathrm{g}}^{(q)} - \boldsymbol{y}_A^{(q)}$.

The inertia takes the old velocity $\boldsymbol{v}_A^{(q)}$ for the calculation of the new velocity $\boldsymbol{v}_A^{(q+1)}$ into account, adding an inertia to the algorithm which prevents too large changes between two iterations. The cognition considers the particle's best position throughout the whole optimization process, whereas the social part contributes the swarm's best position for each update step. According to Shi & Eberhart (1998), the inertia represents the global search properties, whereas the cognition and social part contribute to a local search. The parameters w, ξ_1, and ξ_2 control the influence of each component on the algorithm and are problem specific. Tuning of the parameters allows a switching between exploration (global) and exploitation (local). In general, the convergence behavior of PSO depends highly on the parameter choice and the given problem (Poli et al., 2007). Yet, PSO has been successfully applied to a variety of different design optimization problems (Kennedy & Eberhart, 1995; Poli et al., 2007).

2.3.2 Topology optimization

For the component optimization between hierarchy level (II) and (III), structural design optimization techniques are to be utilized, that are based on high-fidelity FEM models. To cast a given structural design optimization problem into a mathematical optimization formulation, typically three different types of structural design optimization are distinguished (Bendsøe & Sigmund, 2004):

1. sizing,
2. shape, and
3. topology.

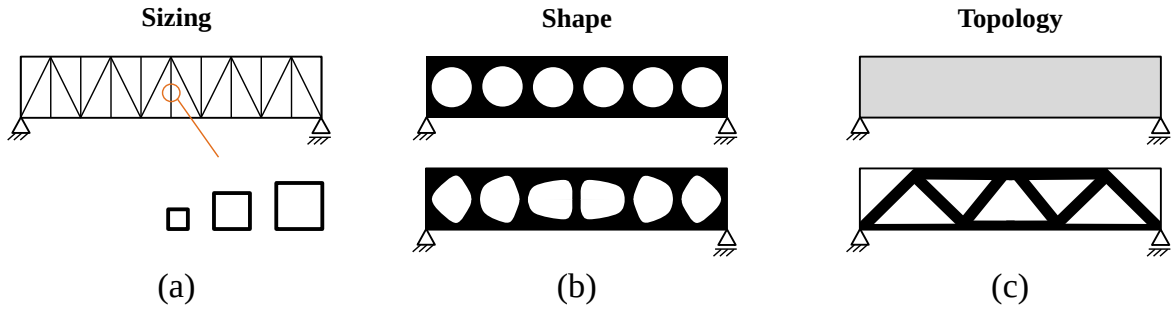


Figure 2.9: Three types of structural design optimization: (a) Sizing, (b) shape, and (c) topology optimization

In a typical sizing problem, the design variables $\boldsymbol{x}$ correlate with an indirect change of geometry, such as the cross-section of a truss element, see Fig. 2.9 (a). Note that the general design domain Ω of a sizing design problem remains the same during optimization and is determined a priori. In contrast, the goal of a shape optimization problem is to find the optimal shape of the design domain Ω. Therefore, the shape itself of the design domain Ω is described by design variables $\boldsymbol{x}$ and are changed during the optimization process. This sometimes affects the underlying FEM model and requires remeshing, complicating the design optimization process. Finally, topology optimization is concerned

with determining the most general features, such as the number, location, and shape of holes and the connectivity of a design domain Ω with given geometrical dimensions (Bendsøe & Sigmund, 2004). Therefore, it is usually the starting point of an structural design optimization task, while shape and/or size optimization is performed afterwards.

In this work, we are concerned with topology optimization, but all developed methods also work with other types of structural design optimization. Topology optimization was first established by Bendsøe & Kikuchi (1988) aiming to solve the following structural design optimization problem:

> How to distribute material in a given design domain Ω to achieve an optimal or best design within the available means.

Since then, several approaches have been developed to solve this design optimization problem, such as density-based approaches or level-set, topological derivative, phase-field, and evolutionary approaches (Sigmund & Maute, 2013). Starting point for all classical topology optimization approaches is a mechanical body in linear elastostatics, as already introduced in Section 2.1, that is subject to boundary conditions and outer loads. If the stiffness is to be maximized, i.e., minimization of the internal energy, the following optimization problem can be formulated

$$\begin{aligned} &\min_{\boldsymbol{d}\in\boldsymbol{D},\boldsymbol{E}} l(\boldsymbol{d}), \\ \text{s. t.:}\quad & a_{\mathrm{E}}(\boldsymbol{d},\delta\boldsymbol{d}) - l(\delta\boldsymbol{d}) = 0, \quad \text{for all } \delta\boldsymbol{d}\in\boldsymbol{D}, \\ & \boldsymbol{E}\in\boldsymbol{E}_{\mathrm{ad}}, \end{aligned} \tag{2.54}$$

where $a_{\mathrm{E}}(\boldsymbol{d},\delta\boldsymbol{d}) = \int_\Omega \delta\boldsymbol{\varepsilon}^\top\boldsymbol{\sigma}\, d\Omega$ and $l(\delta d) = \int_\Omega \delta\boldsymbol{d}^\top\boldsymbol{b}\, d\Omega + \int_\Gamma \delta\boldsymbol{d}^\top\boldsymbol{t}\, d\Gamma$ are the internal and external virtual work components of the static equilibrium equation in the weak form of Section 2.1.1. In topology optimization, $l(\delta d)$ is referred to as compliance C, $\boldsymbol{D}$ refers to the set of kinematically admissible displacement fields, and $\boldsymbol{E}$ are the stiffness values as design variables belonging to the set of admissible stiffness tensors in $\boldsymbol{E}_{\mathrm{ad}}$ for a predefined reference volume V_0. A topology-optimized domain Ω in terms of compliance C for a given load $\boldsymbol{t}$ and a volume V_0 is shown in Fig. 2.10.

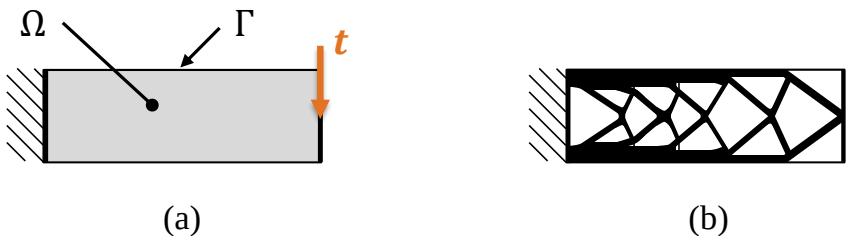


Figure 2.10: (a) A mechanical body Ω in linear elastostatics clamped on the left boundary subject to a traction load $\boldsymbol{t}$ on the right boundary and (b) the respective optimal topology for minimized internal energy, while satisfying a volume constraint

When the problem (2.54) needs to be solved in practice, FEM as a bottom-up mapping is usually utilized. Among the investigated topology optimization methods, the so-called Simplified Isotropic Material with Penalization (SIMP) model is the most common one (Bendsøe & Sigmund, 2004). Here, the stiffness for a given isotropic material is scaled throughout the discretized design domain Ω

$$\rho_e^p \boldsymbol{K}_e, \qquad p \geq 1, \tag{2.55}$$

$$\sum_{e=1}^{n_{\mathrm{ele}}} \rho_e V_e \leq V_0, \quad 0 < \rho_{\mathrm{lb},e} < \rho_e \leq 1, \tag{2.56}$$

where $\rho_e{=}x_j$ are the design variables and are usually referred to as densities in a SIMP-based topology

optimization. The density ρ_e scales the stiffness of each element in Ω between a lower threshold and a reference stiffness $\boldsymbol{K}_e$. In addition, ρ_e also scales the volume per element V_e and hence the total volume V. Since for non-composite materials, intermediate densities $0<\rho_e<1$ have no physical meaning, a discrete 0-1 design is desired. In SIMP, one therefore usually chooses a value of $p>1$, so intermediate densities are unfavorable in the sense that the stiffness achieved is less than the volume V of the material, implicitly directing the optimization to a 0-1 design. Choosing p too low or too high leads to either gray-scale problems, i.e., too many intermediate densities, or too fast convergence to local minima. The value that ensures good convergence to almost 0-1 solutions is $p=3$.

The minimum compliance optimization problem of (2.54) can then be reformulated as

$$\begin{aligned} &\min_{\rho_e} C(\rho_e) = \boldsymbol{f}^\top \boldsymbol{d}, \\ \text{s. t.:}\quad &\textstyle\sum_{e=1}^{n_{\text{ele}}} \rho_e V_e - V_0 \leq 0, \\ &\rho_{\text{lb},e} \leq \rho_e \leq \rho_{\text{ub},e}, \qquad \text{for } e = 1, ..., n_{\text{ele}}, \end{aligned} \tag{2.57}$$

and the FEM equilibrium $(\mathrm{A}_{e=1}^{n_{\text{ele}}} \rho_e^p \boldsymbol{K}_e)\boldsymbol{d} = \boldsymbol{f}$ is maintained within each function call to compute $\boldsymbol{d}$.

Since the number of design variables n_{x} is equal to the number of finite elements n_{ele} of the model, the dimensionality of the optimization problem is usually very high, while the number of constraints is small in comparison. Therefore, gradient-based optimization algorithms are typically utilized to solve (2.57). To compute the derivatives, the adjoint method can be used, in which the derivatives of the displacements $\boldsymbol{d}$ are not computed explicitly. For the minimum compliance problem (2.57) the required gradients can be calculated as follows

$$\frac{\partial C}{\partial \rho_e} = -p\rho_e^{p-1} \boldsymbol{d}^\top \boldsymbol{K}_e \boldsymbol{d}, \tag{2.58}$$

whereas the gradients for the volume are

$$\frac{\partial V}{\partial \rho_e} = V_e. \tag{2.59}$$

Therefore, derivatives for the minimum compliance problem are extremely cheap to compute with only localized derivatives, i.e., each derivative contains only element-level information. However, hidden in the displacement vector $\boldsymbol{d}$ of the compliance gradients is an effect of the other design variables since the system displacement is naturally affected by each element and thus by all design variables ρ_e.

Besides the classical minimum compliance problem in topology optimization, often also other quantities of interests need to be calculated. In the monolithic optimization problem (A) of (1.3), displacement gradients instead of the compliance are needed. Yet, with a small modification also these can be derived via the adjoint method as

$$\frac{\partial d_f}{\partial \rho_e} = -p\rho_e^{p-1} \boldsymbol{\lambda}^\top \boldsymbol{K}_e \boldsymbol{d}, \tag{2.60}$$

where d_f is the displacement of interest of the global displacement vector $\boldsymbol{d}$, and $\boldsymbol{\lambda}^\top$ is the solution of the adjoint load problem $\boldsymbol{K}\boldsymbol{\lambda} = \mathbf{1}$, where $\mathbf{1}$ has only zero entries except for the particular degree of freedom f, which is $1_f = 1$. For more details, see Bendsøe & Sigmund (2004).

The most common algorithm for topology optimization problems is the method of moving asymptotes (MMA) developed by Svanberg (1987). It is a gradient-based algorithm that utilizes a special type of convex approximations based on local information, which are supposed to stabilize and accelerate the convergence of the MMA. In addition, the asymptotes provide an extra degree of flexibility that can be

leveraged to solve a given design problem more effectively. The starting optimization problem for the MMA is

$$\begin{aligned} &\min_{x_j,\upsilon_k,\zeta} f(x_j) + a_0\zeta + \sum_{k=1}^{n_\mathrm{g}} \left(c_k\upsilon_k + \tfrac{1}{2}d_k\upsilon_k^2\right), \\ \text{s. t.:}\quad & g_k(x_j) - a_k\zeta - \upsilon_k \leq 0 && \text{for } k = 1, ..., n_\mathrm{g} \\ & x_{\mathrm{lb},j} \leq x_j \leq x_{\mathrm{ub},j}, && \text{for } j = 1, ..., n_\mathrm{x}, \\ & 0 \leq \upsilon_k \\ & 0 \leq \zeta. \end{aligned} \tag{2.61}$$

For the minimum compliance problem (2.57), the design variables are the element densities x_j=ρ_e, the objective is the compliance $f(x_j)$=C and the inequality constraint is concerned with the volume $g(x_j)$=$\sum_{e=1}^{n_\mathrm{ele}} \rho_e v_e - V_0$. The artificial design variables υ and ζ are computed by the MMA during the optimization. The heuristic parameters are set to a_0=1, a_k=0, and d_k=0, while for c_k it is recommended by Svanberg (2007) to use a reasonable large number, e.g., $10^3 \leq c_k \leq 10^4$.

For each iteration q at point $[\boldsymbol{x}^{(q)}, \boldsymbol{\upsilon}^{(q)}, \zeta^{(q)}]$, a MMA subproblem is constructed

$$\begin{aligned} &\min_{x_j,\upsilon_k,\zeta} \hat{f}(x_j) + a_0\zeta + \sum_{k=1}^{n_\mathrm{g}} \left(c_k\upsilon_k + \tfrac{1}{2}d_k\upsilon_k^2\right) \\ \text{s. t.:}\quad & \hat{g}_k(x_j) - a_k\zeta - \upsilon_k \leq 0, && \text{for } k = 1, ..., n_\mathrm{g} \\ & \alpha_{\mathrm{lb},j} \leq x_j \leq \alpha_{\mathrm{ub},j}, && \text{for } j = 1, ..., n_\mathrm{x}, \\ & 0 \leq \upsilon_k, \\ & 0 \leq \zeta, \end{aligned} \tag{2.62}$$

where $\hat{f}(x_j)$ and $\hat{g}_k(x_j)$ are approximations of the original functions based on first-order information at the current iteration $\boldsymbol{x}^{(q)}$ and the lower and upper moving asymptotes, l_j and u_j, respectively. For more information on the approximations and the correct choice of parameters, see Svanberg (2007).

By constructing the Lagrange function $L(\boldsymbol{x}, \boldsymbol{\upsilon}, \zeta, \boldsymbol{\lambda})$ of the given optimization problem (2.62), the Karush-Kuhn-Tucker optimality conditions can be derived. Using a primal-dual interior point method, the relaxed problem can be converted into a system of linear equations. Taking advantage of the duality and eliminating $\boldsymbol{\upsilon}$, it can be solved for the dual Lagrange multipliers $\boldsymbol{\lambda}$ and the artificial design variable ζ, instead of the primary design variables $\boldsymbol{x}$. This alters the system size according to Svanberg (2007) from

$$\mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}, \tag{2.63}$$

to

$$\mathbb{R}^{(n_\mathrm{g}+1)\times(n_\mathrm{g}+1)}. \tag{2.64}$$

For problems with a high number of design variables n_x, but low number of constraints n_g, this can reduce the system size considerably. Finally, for each iteration q, a new design in the primal space is calculated as

$$[\boldsymbol{x}^{(q+1)}, \boldsymbol{\upsilon}^{(q+1)}, \zeta^{(q+1)}]. \tag{2.65}$$

For any optimal solution, $\boldsymbol{\upsilon}^*$=$\mathbf{0}$ and ζ^*=0 should hold. Even though the MMA relies on some heuristic parameters, the algorithm works very well for a variety of structural design optimization problems (Svanberg, 1987).

Regardless of the optimization algorithm, there are three main problems of a SIMP-based topology optimization (Sigmund & Petersson, 1998; Bendsøe & Sigmund, 2004):

1. mesh-dependency,
2. checkerboard pattern, and
3. non-convexity.

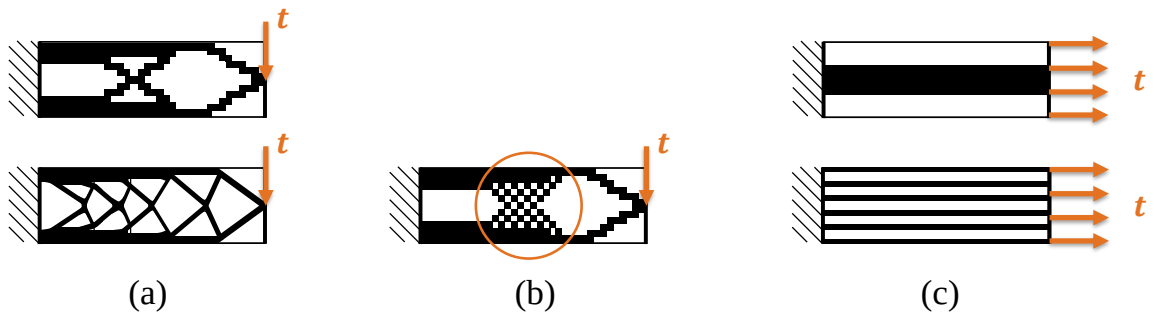


Figure 2.11: (a) Optimization results for a coarse and fine mesh, (b) checkerboard pattern, and (c) non-unique solutions of SIMP-based topology optimization

First, there is a mesh-dependency for results of SIMP-based topology optimization. This is due to the fact that in general there is no unique solution to the continuous mechanical problem (2.54). Introducing new holes while keeping the volume constant always increases the quality of the given structure. Similarly, refining the mesh in a discretized domain Ω leads to a better but also qualitatively different topology, see Fig. 2.11 (a). Ideally, the mesh refinement should only lead to a better description of the boundaries of the topology, but with the same qualitative characteristics.

Second, SIMP-based topology optimization also suffers from so-called checkerboard patterns. Checkerboard patterns involve structures whose densities vary periodically, similar to a checkerboard consisting of alternating fixed (ρ_e=1) and empty elements (ρ_e=$\rho_{\mathrm{lb},e}$), see Fig. 2.11 (b). Although a checkerboard pattern does not lead to a meaningful elastic behavior, such designs are often produced by a topology optimization. One explanation for the appearance of checkerboards in topology optimization is that such material arrangements exhibit artificially high stiffness when analyzed in certain discretized formulations.

For the mesh-dependency as well as checkerboard patterns, various solution procedures do exist, the interesting reader is referred to Sigmund & Petersson (1998) or Bendsøe & Sigmund (2004). Within this work, a gradient sensitivity filter is utilized to solve both problems

$$\frac{\partial \tilde{f}}{\partial \rho_e} = \frac{1}{\rho_e \sum_{i=1}^{n_{\mathrm{ele}}} \tilde{H}_i} \sum_{i=1}^{n_{\mathrm{ele}}} \tilde{H}_i \rho_i \frac{\partial f}{\partial \rho_i}, \tag{2.66}$$

whereas n_{ele} is the total number of elements in the mesh. The mesh-independent convolution operator $\tilde{H}_i$ is defined as

$$\tilde{H}_i = r_{\min} - \mathrm{dist}(e,i), \quad \{i \in n_{\mathrm{ele}} \,|\, \mathrm{dist}(e,i) \leq r_{\min}\}, \quad \text{for } e=1,\ldots,n_{\mathrm{ele}}, \tag{2.67}$$

where $\mathrm{dist}(e,i)$ is the distance between the center of the element e and another element i in the dimensionless parameter space of the finite element mesh. The operator $\tilde{H}_i$ is zero outside the filter radius. For a filter radius of $r_{\min}$=0, the optimization converges to the original solution and for $r_{\min}$=∞ an evenly distributed material is obtained. Unfortunately, the sensitivity filter is a heuristic and the theoretical basis not yet understood. However, computational experience has shown that

filtering of the sensitivity information of the optimization problem is a highly efficient way to ensure mesh-independency and avoid checkerboard patterns.

Third and lastly, structural topology optimization problems for $p>1$ are non-convex, meaning many problems may have multiple optima, i.e., non-unique solutions. An example of this is the design of a uniaxially tensioned structures. Here, a structure consisting of one thick bar is just as good as a structure consisting of several thin bars of the same cross-sectional area, see Fig. 2.11 (c). This non-convexity typically means that one can find several different local minima in gradient-based optimization algorithms, and that one can obtain different solutions to the same discretized problem by choosing different initial start vectors $\boldsymbol{x}^{(0)}$ and different parameters of the utilized optimization algorithm.

After a topology optimization has been performed, considering the previously discussed problems, the results usually must be interpreted, e.g., in terms of a conversion to a CAD file during post-processing. These post-processing steps are necessary tasks within a SIMP-based topology optimization and many different approaches have been proposed to solve this task. For example, the output can be smoothed using image processing tools. Many commercial optimization softwares automatically smooth the outputs, so care must be taken here as the underlying problem, e.g., checkerboard pattern or gray-scale, is ignored (Sigmund & Petersson, 1998). Another approach that emerged soon after the introduction of topology optimization are methods that combine topology optimization with shape optimization in a post-processing step (Sigmund & Maute, 2013). Even if an extra shape optimization is tedious, the fulfillment of the requirements can be explicitly ensured here. In addition to pure post-processing methods, many other topology optimization approaches are originally motivated by the need to provide results that reduce the effort of subsequent post-processing steps compared to SIMP-based topology optimization in the first place (Deaton & Grandhi, 2014).

3 Distributed design optimization

3.1 Introduction

Distributed design optimization originates from a research area called *multidisciplinary design optimization* (MDO). It investigates the application of numerical optimization techniques to the design of engineering systems that span multiple disciplines or components (Martins & Lambe, 2013). In addition to classical high-fidelity bottom-up mappings, such as FEM, also low-fidelity meta models are often utilized here. Formulating the design problem and coordinating different bottom-up mappings is a critical part within MDO. The combination of a coordination strategy and one or more problem formulations is called an *architecture*. This architecture can be either *monolithic* or *distributed*, and the right choice can significantly reduce the computational time required to solve the given optimization problem. Monolithic architectures form and solve a single optimization problem, although several bottom-up mappings may be included. In contrast, distributed optimization architectures decompose the single optimization problem into a set of smaller optimization subproblems containing subsets of the objectives, design variables, and constraints (Martins & Lambe, 2013). The decomposition itself can either be done along discipline boundaries, which is called *aspect partitioning* and is the classical area of MDO. Another possibility is *object partitioning*, where physical components are decomposed (Tosserams et al., 2009). Since we focus on object partitioning, our different subproblems are now referred to as top-level *system optimization* and lower-level *component optimization*. However, the component optimizations could theoretically also be different disciplines. There are two main reasons to prefer distributed optimization architectures over monolithic ones. The first reason is to mimic the classical product development process where groups design their tasks independently, making distributed optimization architectures supposedly easier to adopt in industry. The second is the ability to decompose the problem to reduce computational time (Martins & Ning, 2022). Among distributed architectures, there are several properties that need to be considered to apply the right architecture to the given problem. In this work, we distinguish between distributed architectures that possess

1. a decoupled vs. nested vs. alternating formulation,
2. a hierarchical vs. non-hierarchical decomposition,
3. a parallel vs. non-parallel execution, and those that
4. converge vs. not converge to the monolithic solution.

According to Tosserams et al. (2009), *nested* formulations are two-level programming problems in which the lower-level component optimizations are nested within a coordinating system optimization. In contrast, *alternating* formulations iterate between solving a system optimization and the respective component optimizations. In addition, we introduce a third *decoupled* formulation that is executed strictly sequentially, i.e., first on the system level and then on the lower component levels, without requiring iteration back to the system level or coordination among the different components. Depending on the formulation, the degree of dependency and thus the coordination effort required to solve the problem decreases from nested to alternating to decoupled. Furthermore, the type of decomposition of the monolithic optimization problem can differ between *hierarchical* or *non-hierarchical* (Papalambros & Wilde, 2018). In this work, a hierarchical architecture has a system optimization and a component optimization, and each optimization subproblem operates within clear hierarchical levels, e.g., the system optimization problem should not deal with detailed design variables $\boldsymbol{x}$ of hierarchy level (III). Moreover, whether a *parallel* or a *non-parallel* execution is implemented can help reduce the computational time of large-size optimization problems. Finally, a central concern of distributed optimization architectures is that they converge to the *monolithic solution* of their monolithic counterpart (Martins & Lambe, 2013; Martins & Ning, 2022).

3.2 Distributed optimization architectures

In general, there is a wide variety of distributed optimization architectures, see, e.g., Tosserams et al. (2009) or Martins & Lambe (2013). In the following chapter only the distributed architectures most relevant for the given monolithic optimization problem (A) of Section 1.2 are discussed. The notation is therefore adapted to a complicating constraints problem with a separable objective function $f(\boldsymbol{x})$, a system-wide constraint function $g_{(0)}(\boldsymbol{x})$, no shared design variables $\boldsymbol{x}_{(0)}$, and three distinct hierarchy levels $[\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{x}]$. Hierarchy levels (II) and (III) are concerned only with their local design variables $\boldsymbol{x}_{(i)}$ and responses $\boldsymbol{y}_{(i)}$. For simplicity, the equality constraints $\boldsymbol{h}(\boldsymbol{x})$=$\boldsymbol{0}$ and the upper and lower bounds of the respective design variables are neglected in this chapter.

3.2.1 Individual optimization

An intuitive approach that is often used in industry is an individual optimization (IO). Here, a concurrent design of the entire system is often not possible due to the difficulty of coordinating all departments and their respective engineers responsible for a particular component or discipline. Hence, each component i of the system is optimized individually with respect to its own local objective function, design variables, and constraints

$$\begin{aligned} & \min_{\boldsymbol{x}_{(i)}} f_{(i)}(\boldsymbol{x}_{(i)}), \\ \text{s. t.:} \quad & \boldsymbol{g}_{(0)}(\boldsymbol{x}_{(i)}) \leq \boldsymbol{0}, \\ & \boldsymbol{g}_{(i)}(\boldsymbol{x}_{(i)}) \leq \boldsymbol{0}. \end{aligned} \tag{3.1}$$

One can perform the IO of (3.1) either sequentially, see Fig. 3.1 (a), in which case it is also called sequential optimization (Martins & Ning, 2022), or in parallel, see Fig. 3.1 (b), in which case some assumptions are made to decouple the problems, see, e.g., Krischer et al. (2020).

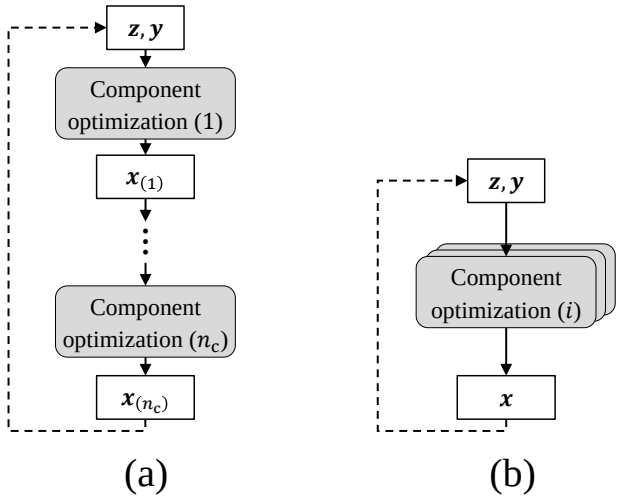


Figure 3.1: Flowchart of the individual optimization architecture (IO) in (a) the classical sequential way and (b) in parallel for specific assumptions made beforehand

In both cases, no interdependencies are taken into account and no system optimization takes place to steer all individual components of the system toward the optimal overall system design. Instead, after each step, the respective results $\boldsymbol{x}_{(i)}$ are simply passed on and the system behavior is evaluated at the end (Kroo, 1997; Martins & Ning, 2022).

The ease of implementation without the need for direct interaction between the different bottom-up mappings makes this approach attractive. Since the component optimization problems (3.1) do not need to consider coupling or consistency, the dimensionality of each optimization problem is not increased and can be solved with the classical available algorithms, while parallelization is possible if decoupling is done beforehand, see Fig. 3.1 (b). However, optimal solutions cannot usually be found, and even feasible solutions require a large number of outer iterations, so the whole process often has to be run several times. Since an IO does not take into account mutual dependencies and does not converge to the original problem solution of a monolithic optimization, it is often not considered a real distributed architecture (Martins & Ning, 2022).

3.2.2 Collaborative optimization

One of the most mature distributed architectures is collaborative optimization (CO) conceived by Braun (1996). The CO architecture is inspired by the way different design teams work together in the context of complex engineering systems in industry. In CO, the system optimization problem is responsible for minimizing the system objective, while the component optimizations minimize system inconsistencies (Braun et al., 1996; Braun & Kroo, 1997).

The system optimization problem is

$$\begin{aligned} & \min_{\tilde{\boldsymbol{x}}_{(i)}, \boldsymbol{y}^t_{(i)}} \; f(\tilde{\boldsymbol{x}}_{(i)}, \boldsymbol{y}^t_{(i)}), \\ \text{s. t.:} \quad & \boldsymbol{g}_{(0)}(\tilde{\boldsymbol{x}}_{(i)}, \boldsymbol{y}^t_{(i)}) \leq \boldsymbol{0}, \\ & J^*_{(i)} = ||\tilde{\boldsymbol{x}}_{(i)} - \boldsymbol{x}_{(i)}||_2^2 + ||\boldsymbol{y}^t_{(i)} - \boldsymbol{y}_{(i)}||_2^2 = 0 \qquad \text{for } i = 1, ..., n_\mathrm{c}, \end{aligned} \tag{3.2}$$

where the constraint function $J_{(i)}$ measures the consistency between requested system-level values and the component-level values. The nested component optimization subproblem is then

$$\begin{aligned} & \min_{\boldsymbol{x}_{(i)}} J_{(i)}(\boldsymbol{x}_{(i)}), \\ \text{s. t.:} \quad & \boldsymbol{g}_{(i)}(\boldsymbol{x}_{(i)}) \leq \boldsymbol{0}. \end{aligned} \tag{3.3}$$

The general procedure of the CO can be seen in Fig. 3.2. CO is a nested architecture, meaning each system optimization iteration includes n_c complete component optimizations.

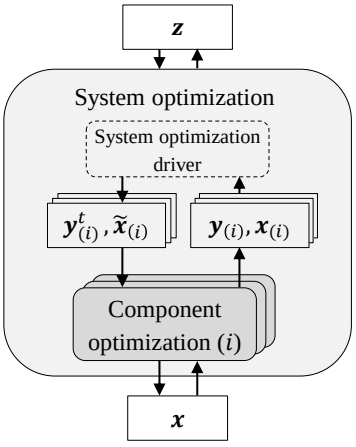


Figure 3.2: Flowchart of the collaborative optimization architecture (CO)

The system optimization driver provides target values $\boldsymbol{y}^t_{(i)}$ and copies of the component's design variables $\tilde{\boldsymbol{x}}_{(i)}$ to the component optimizations i. Each component optimization is carried out independently of each other and returns the computed response $\boldsymbol{y}_{(i)}$ as well as the component's design variables $\boldsymbol{x}_{(i)}$ to the system optimization, after each component optimization converged.

The complete independence of the component optimization problems combined with a simple coordination pattern makes this architecture attractive for weakly coupled design problems. Due to the nested formulation, post-optimization sensitivity analysis, i.e., computing derivatives with respect to an optimized function, is required which complicates the evaluation of the consistency constraint functions $J_{(i)}$. Since the system optimization uses local copies $\tilde{\boldsymbol{x}}_{(i)}$ of the design variables of the component optimization problems, the dimensionality of CO can increase significantly depending on the problem formulation, e.g., when topology optimization is used. These local copies also prevent CO from being a strictly hierarchical architecture. However, copies of the local design variables are only made when these variables directly affect the objective. The component optimizations within the CO can be run in parallel. Moreover, it has been proved that the CO problem converges to the original solution of a monolithic optimization. However, it shows problems in terms of numerical performance due to the underlying mathematical formulation. Among the observed problems are infeasible results as well as severe convergence issues leading to a higher number of iterations or no convergence at all (Martins & Lambe, 2013).

3.2.3 Analytical target cascading

Analytical target cascading (ATC) was developed by H. M. Kim et al. (2003) as a hierarchical multi-level coordination strategy to support the top-down development processes in the automotive industry. The goal is to turn top-level system targets into detailed component-level specifications while also enabling parallel design activities. On each level, each subproblem tries to meet its own objectives and target values, if existing, while passing on own responses and targets to other levels. This negotiation between upper-level requirements and lower-level resources takes place at every level until all subproblems converge (Martins & Lambe, 2013; Papalambros & Wilde, 2018).

The system optimization formulation is

$$\begin{aligned} \min_{\boldsymbol{y}^t_{(i)}} \; & \textstyle\sum_{i=1}^{n_\mathrm{c}} f_{(i)}(\boldsymbol{y}^t_{(i)}) + \sum_{i=1}^{n_\mathrm{c}} \quad P_{(i)}\left(\boldsymbol{y}^t_{(i)} - \boldsymbol{y}_{(i)}\right), \\ \text{s. t.:} \quad & \boldsymbol{g}_{(0)}(\boldsymbol{y}^t_{(i)}) \leq \boldsymbol{0}, \qquad \text{for } i = 1, ..., n_\mathrm{c}, \end{aligned} \tag{3.4}$$

whereas the component optimization subproblems are defined as

$$\begin{aligned} \min_{\boldsymbol{x}_{(i)}} \; & f_{(i)}(\boldsymbol{x}_{(i)}) + P_{(i)}\left(\boldsymbol{y}^t_{(i)} - \boldsymbol{y}_{(i)}\right), \\ \text{s. t.:} \quad & \boldsymbol{g}_{(i)}(\boldsymbol{x}_{(i)}) \leq \boldsymbol{0}. \end{aligned} \tag{3.5}$$

The general ATC procedure is shown in Fig. 3.3. ATC is an alternating formulation with two iteration loops. For each inner iteration of the distributed architecture, the system optimization is solved first and the target values $\boldsymbol{y}^t_{(i)}$ are passed to each component optimization problem. Each component optimization problem can be solved in parallel and computes its respective responses $\boldsymbol{y}_{(i)}$ based on $\boldsymbol{x}_{(i)}$. Consistency between the system and component optimizations is ensured by a relaxed penalty formulation $P_{(i)}$ in the objective function at each level. In the outer iteration q, the weights of the penalty function $P_{(i)}$ are updated until the convergence criteria are satisfied.

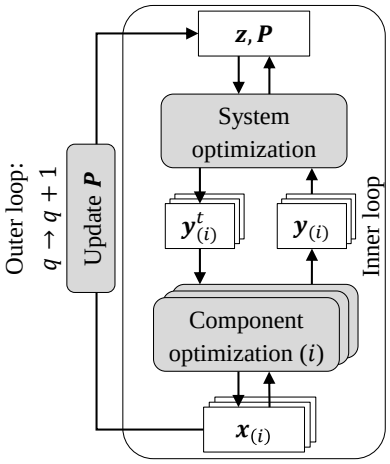


Figure 3.3: Flowchart of the analytical target cascading architecture (ATC)

Often, ATC and the earlier CO architecture are seen as related architectures sharing the same basic idea with a simple coordination pattern between the different problem levels. However, unlike the nested formulation of CO, the alternating formulation of ATC simplifies the implementation, e.g., no post-optimization sensitivity analysis is required when computing gradients. Moreover, the original ATC formulation is strictly hierarchical. For convex problems, there is a mathematical proof of convergence to the optimum of a monolithic optimization (Han & Papalambros, 2010). However, relaxation via penalty functions $P_{(i)}$ allows for inconsistencies in the convergent solution $\boldsymbol{x}^*$ and ATC is therefore often rather used in the early stages of product development to check whether the set goals are achievable at all. Moreover, the choice of penalty functions $P_{(i)}$ and the actual magnitude of the weights during the optimization have a large impact on the results. There are no strict rules in this regard, and either quadratic formulations or an augmented Lagrangian formulation are recommended (H. M. Kim et al., 2003; Tosserams et al., 2006).

3.2.4 BLISS-2000

The original bilevel integrated system synthesis (BLISS) was developed by Sobieszczanski-Sobieski et al. (2000) and follows the idea of decomposing engineering design problems along disciplinary or component lines. In Sobieszczanski-Sobieski et al. (2003), an enhancement of the original approach was introduced called BLISS-2000, where the system optimization is carried out on meta models, and the component optimizations are fully autonomous.

For BLISS-2000, the system optimization problem is given by

$$\begin{aligned} &\min_{\boldsymbol{y}^t_{(i)}, \boldsymbol{w}_{(i)}} \; f(\boldsymbol{y}^t_{(i)}), \\ \text{s. t.:} \quad &\boldsymbol{g}_0(\boldsymbol{y}^t_{(i)}) \leq \boldsymbol{0}, \\ &\boldsymbol{y}^t_{(i)} - \hat{\boldsymbol{y}}_{(i)}(\boldsymbol{w}_{(i)}) = \boldsymbol{0}, \quad \text{for } i = 1, ..., n_\mathrm{c}, \end{aligned} \tag{3.6}$$

and the component optimization is

$$\begin{aligned} &\min_{\boldsymbol{x}_{(i)}} \boldsymbol{w}_{(i)}\, \boldsymbol{y}^\top_{(i)} \left(\boldsymbol{x}_{(i)}\right), \\ \text{s. t.:} \quad &\boldsymbol{g}_{(i)}(\boldsymbol{x}_{(i)}) \leq \boldsymbol{0}. \end{aligned} \tag{3.7}$$

The BLISS-2000 is an alternating distributed optimization architecture, see Fig. 3.4. In each iteration, first a DoE is carried out that trains meta models $\hat{\boldsymbol{y}}_{(i)}$ based on the formulated component optimizations (3.7). These meta models are then used to compute the responses for the system optimization (3.6). The system optimization problem determines the target values $\boldsymbol{y}^t_{(i)}$ and the weight coefficients $\boldsymbol{w}_{(i)}$ that are related to the component optimization objectives. Consistency between the target values $\boldsymbol{y}^t_{(i)}$ and the approximated responses of the meta models $\hat{\boldsymbol{y}}_{(i)}$ are considered by the consistency constraint $\boldsymbol{y}^t_{(i)} - \hat{\boldsymbol{y}}_{(i)} = \boldsymbol{0}$ on the system level. The weights $\boldsymbol{w}_{(i)}$ can be interpreted as a measure of control over the respective component optimizations (Martins & Lambe, 2013).

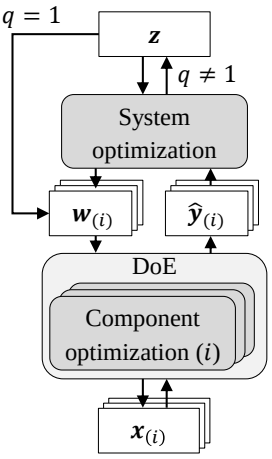


Figure 3.4: Flowchart of the BLISS-2000 architecture

Even though BLISS-2000 contains a DoE, it has a relatively simple alternating solution procedure. The strictly hierarchical formulation also keeps the number of design variables at each level relatively low by adding targets $\boldsymbol{y}^t_{(i)}$ and weights $\boldsymbol{w}_{(i)}$ only at the system level. As the meta model approximation is burdened with an error, iterative refitting of the meta models with a DoE needs to be conducted in each iteration. Since the meta models are trained for each component, and the problems are independent, the calculations can be run in parallel with minimal coordination. If the fully parallel potential is exploited, BLISS-2000 has the capability of outperforming other distributed architectures (Martins & Lambe, 2013). Moreover, Sobieszczanski-Sobieski et al. (2003) proved that BLISS-2000 is equivalent to a monolithic optimization, if the problem is convex.

3.2.5 Quasiseparable decomposition

The quasiseparable decomposition (QSD) was proposed by Haftka & Watson (2005) for a particular class of optimization problems that are narrow enough for a rigorous decomposition theory, yet general enough to encompass the majority of large-scale engineering design problems. The decomposition approach itself is inspired by rounds of negotiations in a company between different hierarchical levels and is therefore intended to represent a classical product development process in industry.

The system optimization problem reads

$$\begin{aligned} &\min_{\boldsymbol{y}^t_{(i)}, b_{(i)}} && f(\boldsymbol{y}^t_{(i)}) + \textstyle\sum_{i=1}^{n_\mathrm{c}} b_{(i)}, \\ &\text{s. t.:} && \boldsymbol{g}_{(0)}(\boldsymbol{y}^t_{(i)}) \leq \boldsymbol{0}, \\ &&& -s^*_{(i)}(\boldsymbol{y}^t_{(i)}, b_{(i)}) \leq 0, \qquad \text{for } i = 1, ..., n_\mathrm{c}, \end{aligned} \tag{3.8}$$

while the component optimization is

$$\begin{aligned} & \min_{\boldsymbol{x}_{(i)}, s_{(i)}} -s_{(i)}, \\ \text{s. t.:} \quad & g_{(i)}(\boldsymbol{x}_{(i)}) + s_{(i)} \leq 0, \\ & f_{(i)}(\boldsymbol{x}_{(i)}) + s_{(i)} + b_{(i)} \leq 0, \\ & \boldsymbol{y}^t_{(i)} - \boldsymbol{y}_{(i)}(\boldsymbol{x}_{(i)}) = \boldsymbol{0}. \end{aligned} \tag{3.9}$$

The original formulation is a nested architecture where the solutions $s_{(i)}$ of the component optimization problems (3.9) are constraints in the system optimization. The basic idea is to assign to each component optimization i budgets $b_{(i)}$ and target values $\boldsymbol{y}^t_{(i)}$ from the system level. The component optimizations then independently maximize their constraint margin $s_{(i)}$ while satisfying their objective function $f_{(i)}$ and constraints $g_{(i)}$ under the given budget $b_{(i)}$. The margins $s^*_{(i)}$ are then returned to the system optimization and are used to adjust the values of $\boldsymbol{y}^t_{(i)}$ and the budgets $b_{(i)}$. This decomposition procedure can be viewed as a negotiation between the manager (system level) and the team or department leaders (component level), where the manager presents each with a set of performance requirements and budgets. For each set, the department leader provides a measure of how well they can meet the requirements and budget. In this way, the manager can intelligently allocate resources between teams.

In addition to the original nested procedure of Haftka & Watson (2005), B. Liu et al. (2004) also used QSD in a decoupled formulation, by establishing pre-trained meta models

$$s^*_{(i)} \approx \hat{\boldsymbol{s}}_{(i)}(\boldsymbol{y}^t_{(i)}, b_{(i)}), \tag{3.10}$$

using sample data from multiple component optimizations (3.9). The meta models $\hat{\boldsymbol{s}}_{(i)}$ can then estimate the optimal constraints $s^*_{(i)}$ of the system optimization problem (3.8). Afterwards, n_c component optimizations are performed, which are decoupled and need to be performed only once after the system optimization, see Fig. 3.5.

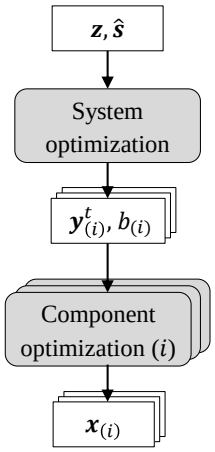


Figure 3.5: Flowchart of the quasiseparable decomposition architecture (QSD) according to B. Liu et al. (2004)

The actual implementation of QSD in the original formulation is complicated due to the gradient computation of the nested and non-continuous component optimization derivatives, so meta models are usually used here as well. In the formulation of B. Liu et al. (2004), the implementation is further simplified by running the approach purely hierarchically and sequentially, which is made possible by

the decoupling after the system optimization. Due to the decoupling also the component optimizations for the training and the actual design process can be run in parallel. A general convergence theory for quasiseparable optimization with continuous variables and C^2-continuity showed that the distributed architecture of QSD can enable global convergence. However, problems were encountered when performing the decoupled optimization architecture. Due to inaccuracies in the meta models, small violations of the constraints could be found after the final component optimizations demanding manual modification of the results (B. Liu et al., 2004).

3.3 Research gap

In the previous section, established distributed optimization architectures were presented that are best suited for the decomposition of the monolithic optimization problem (A) of (1.3). It is interesting to note that all architectures are motivated by the needs of product development processes in industry, however, each single architecture differs in the way of how the mathematical formulation and coordination architecture is set up. In general, distributed architectures have been successfully applied to a variety of use cases but have so far failed to meet the general expectation of enabling adoption in industry by resembling industrial development processes and reducing computational time (Martins & Ning, 2022). One reason is that the decomposed optimization problems of most distributed architectures possess a coordination strategy to maintain consistency. In an industrial context, this coordination would require an optimization architecture that connects all the departments involved, which is difficult to implement. Second, using such a coordination strategy runs the risk of coordination overhead, which makes the entire distributed optimization computationally expensive.

Chapter 1 motivated the need for a decoupled and hierarchical optimization architecture that is based on parallel component optimizations and solves the original monolithic optimization problem (A) using meta models for optimality (regression) and physical feasibility (classification). On the one hand, it is assumed that decoupling, i.e., decomposition without subsequent necessary coordination, can reduce computational effort and avoid coordination overhead. On the other hand, decoupled architectures eliminate the need for an overarching software architecture. Thus, decoupling circumvents several implementation obstacles in practice. Furthermore, a hierarchical formulation resembles the classical product development process in industry, where requirements are typically derived from the management level down to the departments. This is also advantageous from a computational point of view since fewer design variables are available at the system level and the higher-dimensional detail level of the components is decoupled. Decoupling also allows parallel execution, which is another potential for reducing computational time. Finally, convergence to the monolithic solution is a necessary property for distributed optimization architectures. Therefore, for the proposed decoupled architecture, information about optimality and physical feasibility should be provided to the system level by meta models. However, meta model training should not be part of the optimization architecture, but pre-trained meta models are stored in an offline database.

Table 3.1 provides an overview of the presented distributed architectures, evaluating whether an architecture possesses a decoupled formulation (as opposed to nested or alternating), hierarchical decomposition, parallel execution, and converges to the monolithic solution. In addition, the respective bottom-up mappings that are explicitly required for the distributed architectures are presented, where it is distinguished between high-fidelity and low-fidelity bottom-up mappings. In the following a short summary of the presented distributed optimization architectures is given and the challenges in the application of the existing methods to the monolithic optimization problem (A) of (1.3) are outlined.

Table 3.1: Overview of the presented distributed optimization architectures: Individual optimization (IO), collaborative optimization (CO), analytical target cascading (ATC), BLISS-2000, and quasiseparable decomposition (QSD)

Distributed optimization architectures	Architecture properties				Bottom-up mappings		
	Decoupled formulation	Hierarchical decomposition	Parallel execution	Monolithic solution	High-fidelity	Low-fidelity	
						Regression	Classification
IO	x		x		x		
CO			x	x	x		
ATC		x	x	x	x		
BLISS-2000		x	x	x	x	x	
QSD	x	x	x	x	x	x	

IO is often used in industry due to its simplicity. However, it is not considered a real distributed architecture because it generally does not converge to the optimum of the original monolithic problem and is therefore also not suitable for the given design task (A).

In contrast, CO is a distributed architecture, since it has been proved that it converges to the monolithic solution. Moreover, CO allows a parallel working environment for the respective component optimizations. However, it is a nested and also not purely hierarchical formulation, which makes it difficult to apply in the original formulation to high dimensional topology optimization design problems.

The related ATC is a purely hierarchical multi-level approach in an alternating formulation. Similar to CO, it allows for parallel design and equivalence to the monolithic solution can be shown for convex problems. From the main idea, ATC is well suited for the given monolithic optimization problem (A), only the alternating formulation consisting of two loops, an inner and an outer one, might cause coordination overhead and an overarching software architecture is still mandatory.

BLISS-2000 follows the idea of providing component level information to the system level by meta models in a hierarchical and alternating formulation. The system optimization is carried out on meta models, while the component optimizations are executed in parallel. It was shown that it converges to the monolithic solution for convex problems. In contrast to our objective, the expensive training process for the meta models is however part of the optimization architecture and needs to be carried out for each iteration.

Finally, QSD also utilizes meta models. In the original nested formulation rather for sensitivity information, but B. Liu et al. (2004) also used them to establish a decoupled architecture. Here, the training process of the meta models is excluded from the actual design process, making the optimization itself faster. After the system optimization, the component optimizations are completely independent, i.e., decoupled, and can therefore also be run in parallel. However, due to inaccuracies in the meta models, constraints for the final design were violated, hence being physically infeasible. A classifier that evaluates physical feasibility could have helped here to solve this problem. It is also not clear if and how meta models that have already been trained can be reused, which contradicts our general idea of an offline database.

In conclusion, no distributed optimization architecture could be identified that satisfies all our posed requirements to the architecture properties and bottom-up mappings listed in Table 3.1. Nevertheless, partial aspects of the existing and established distributed architectures can be used as a starting point for the distributed architecture under development. Parallelization is a main lever for reducing computational time and has been used in each distributed architecture. A purely hierarchical view was taken by ATC, BLISS-2000, and QSD in accordance with the goal of this work. Finally, both BLISS-2000 and QSD use meta models for the transfer of information about optimality from the component level to the system level, yet lacking estimations on physical feasibility.

In the following chapter, a newly developed hierarchical decoupled optimization architecture (B) based on meta models of an offline database, estimating optimality and physical feasibility, is presented. In addition, a parallel IO formulation (C) is used in Chapter 5 and an ATC formulation (D) in Chapter 6.

4 Informed Decomposition

To decompose the given monolithic optimization problem (A) of (1.3), the hierarchical and decoupled Informed Decomposition (B) was first introduced in Krischer & Zimmermann (2021) and further developed in Krischer et al. (2022). It is divided into

(a) a decoupled optimization architecture consisting of a surrogate-based system optimization that decouples the problem and subsequent independent and parallel component optimizations, and
(b) an offline database consisting of feasibility estimators $\hat{p}$ and mass estimators $\hat{m}$,

see also Fig. 4.1. The decoupled optimization architecture (a) does not need any coordination between the system optimization and the component optimization problems and is based on object partitioning, meaning that the problem is decomposed with respect to the physical components. Since the architecture is decoupled, meta models $\hat{p}$ and $\hat{m}$ are needed to ensure physical feasibility p and optimality with respect to mass m. Note that the process of establishing new meta models (b) is not supposed to be carried out for each new design problem. The idea is, to create an offline database of available meta models that can be used for the majority of design problems. Established meta models can in general be reused for different design problems and have a larger applicability due to similitude methods and explicit consideration of geometrical parameters. If appropriate models do not exist, the procedure (b) is executed to extend the offline database.

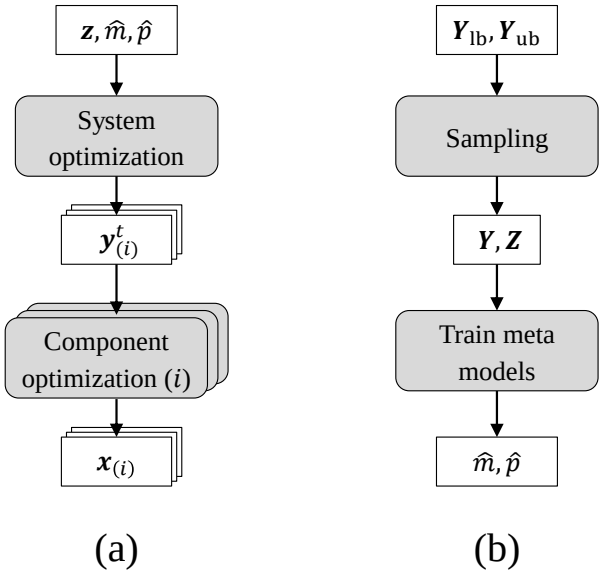


Figure 4.1: Overview of the proposed approach (B): (a) a decoupled optimization architecture consisting of system optimization and decoupled component optimizations, and (b) establishing an offline database of feasibility estimators $\hat{p}$ and mass estimators $\hat{m}$

In the following sections, an alternative representation of interface stiffness matrices $\boldsymbol{K}=\boldsymbol{\Phi}(\boldsymbol{\kappa})$ is introduced. Then, the actual decoupled optimization architecture is presented, which consists of surrogate-based system optimization and decoupled component optimizations. Afterwards, a procedure for building an offline database with a specific active-learning undersampling strategy is explained to obtain well-balanced and efficiently sampled training data. In the last section, the design problem classes are introduced that are utilized to validate the developed approach.

4.1 κ-representation of interface stiffness matrix K

The low-dimensional representation $\boldsymbol{\kappa}$ of the interface stiffness matrix $\boldsymbol{K}$ is a key element of the proposed decoupled optimization architecture. On the one hand, $\boldsymbol{\kappa}$ is used as a target value $\boldsymbol{y}^t$ and is therefore the link between the system and component optimization. On the other hand, $\boldsymbol{\kappa}$ is from importance as it removes redundancies in $\boldsymbol{K}$ and hence reduces the number of design variables of the system optimization and also the number of constraints in the subsequent component optimization. Additionally, it is the input for the meta models $\hat{p}$ and $\hat{m}$. Therefore, first a kernel based on the rigid body modes $\boldsymbol{\phi}_r$ of $\boldsymbol{K}$ is introduced, which establishes the basic mapping $\boldsymbol{\Phi}(\boldsymbol{\kappa})=\boldsymbol{K}$. The size of $\boldsymbol{\kappa}$ can be further reduced by enforcing symmetry conditions on the detailed design $\boldsymbol{x}$ of the components. Finally, the scaling of components based on the κ-representation using similitude methods and explicit geometrical parameters is explained. This extends the scope of the offline database by using same $\hat{p}$ and $\hat{m}$ for $\boldsymbol{\kappa}$ of different geometrical dimensions of the design domain Ω of the structural element.

4.1.1 Kernel of interface stiffness matrix K

For an arbitrary three-dimensional mechanical body with n_{int}=2 interfaces and n_{dof}=6 degrees of freedom per interface, see Fig 4.2, the interface stiffness matrix $\boldsymbol{K}$ determines the elastic behavior of the structural element with respect to the interfaces $\boldsymbol{a}$ and $\boldsymbol{b}$

$$\boldsymbol{K} \in \mathbb{R}^{12\times 12}. \tag{4.1}$$

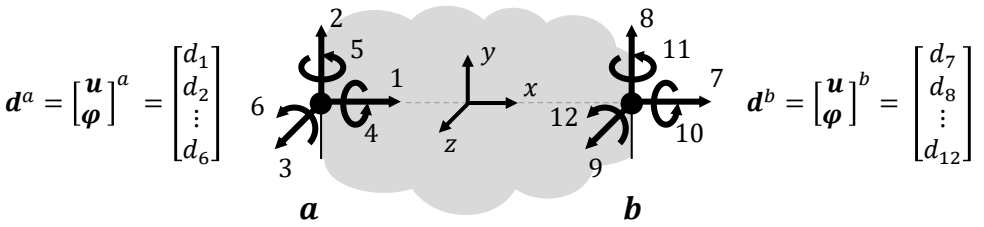


Figure 4.2: A three-dimensional mechanical body with n_{int}=2 interfaces and n_{dof}=6 degrees of freedom per interface

Each interface possesses n_{dof}=6 degrees of freedom , three translational and three rotational, with the corresponding displacements, $\boldsymbol{u}$ and $\boldsymbol{\varphi}$, respectively

$$\boldsymbol{d}^a = [\boldsymbol{u}, \boldsymbol{\varphi}]^{a\top} = \quad [d_1, d_2, ..., d_6]^\top, \quad \in \mathbb{R}^{6\times 1}, \tag{4.2}$$
$$\boldsymbol{d}^b = [\boldsymbol{u}, \boldsymbol{\varphi}]^{b\top} = \quad [d_7, d_8, ..., d_{12}]^\top, \quad \in \mathbb{R}^{6\times 1}. \tag{4.3}$$

Moreover, for the following investigations, the two mechanical interfaces $\boldsymbol{a}$ and $\boldsymbol{b}$ are fixed on the local x-axis with same orientation

$$\boldsymbol{b} - \boldsymbol{a} = [l, 0, 0]^\top. \tag{4.4}$$

From Section 2.1.1 it is known, that feasible interface stiffness matrices $\boldsymbol{K}$ must among others satisfy the following three requirements

(R1) $\boldsymbol{K}$ must be symmetric, i.e., $\boldsymbol{K} = \boldsymbol{K}^\top$,
(R2) rigid body modes $\boldsymbol{\phi}_r$ result in zero forces, i.e., $\boldsymbol{K}\boldsymbol{\phi}_r = \boldsymbol{0}$,
(R3) $\boldsymbol{K}$ must be positive semi-definite, i.e., $\boldsymbol{d}^\top\boldsymbol{K}\boldsymbol{d} \geq 0$.

Additionally, a fourth requirement is imposed demanding that

(R4) there exists a detail design $\boldsymbol{x}$ corresponding to $\boldsymbol{K}$.

Thus, not all interface stiffness matrices $\boldsymbol{K}$ can be realized. If the requirements (R1)-(R4) are all satisfied, $\boldsymbol{K}$ is said to be *physically feasible*.

Due to the symmetry requirement (R1) on the entries $k_{i,j}$ of $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ the following condition holds

$$k_{i,j} = k_{j,i}, \tag{4.5}$$

hence the number of independent stiffness entries reduces from n_k=144 to n_k=78.

Moreover, the mechanical body with n_int=2 interfaces and n_dof=6 degrees of freedom per interface possesses n_ϕ=6 rigid body modes $\boldsymbol{\phi}_r$, see Fig. 4.3, corresponding to three translational and three rotational rigid body modes without any internal deformations. Based on the requirement (R2) the rigid body modes $\boldsymbol{\phi}_r$ can be used to remove dependent stiffness entries of the interface stiffness matrix $\boldsymbol{K}$ by solving the following equation

$$\boldsymbol{K}\boldsymbol{\phi}_r = \boldsymbol{0}, \qquad \text{for } r = 1, \ldots, 6. \tag{4.6}$$

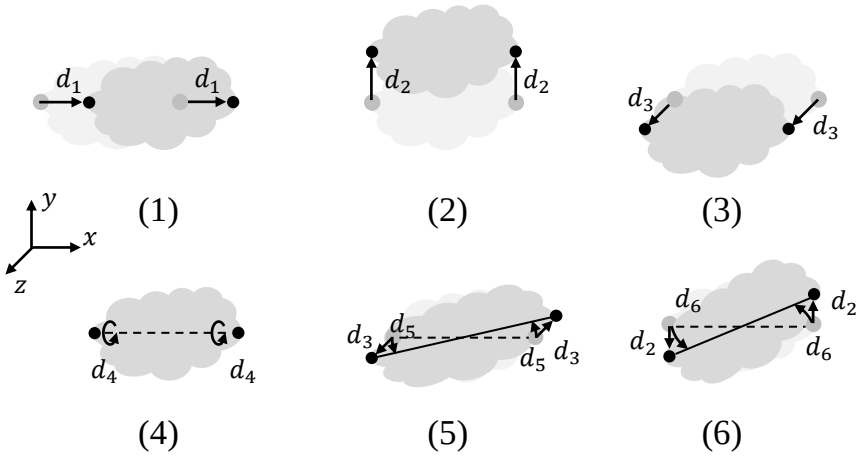


Figure 4.3: n_ϕ=6 rigid body modes of a mechanical body with n_{int}=2 interfaces and n_{dof}=6 degrees of freedom. (1-3) show the translational rigid body modes, while (4-6) show the rotational rigid body modes

From a mathematical point of view these six rigid body modes $\boldsymbol{\phi}_r$ correspond to six eigenvectors with eigenvalues of zero λ_r=0 of the interface stiffness matrix $\boldsymbol{K}$. Equation (4.6) comprises the six-dimensional null space or also called kernel of the interface stiffness matrix $\boldsymbol{K}$. In order to assess the number of independent entries of $\boldsymbol{K}$ the rank-nullity theorem can be used (Meyer, 2008). Here, the rank of $\boldsymbol{K}$ can be calculated as the difference between the dimension of the vector space of $\dim(\boldsymbol{K})$=12 and the dimension of the null space, also called the $\mathrm{nullity}(\boldsymbol{K})$=$n_\phi$=6,

$$\mathrm{rank}(\boldsymbol{K}) + \mathrm{nullity}(\boldsymbol{K}) = \dim(\boldsymbol{K}). \tag{4.7}$$

It follows that $\mathrm{rank}(\boldsymbol{K})$=6 and thus the dimension of the non-degenerated vector space, i.e., the vector space where the linear dependencies are removed, is also $\mathbb{R}^6$. From a mechanical point of view, this can be shown more easily with a small example, see Fig. 4.4.

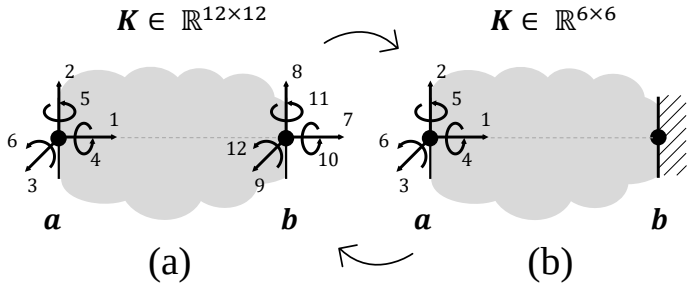


Figure 4.4: A three-dimensional mechanical body that is (a) freely moving in space without support $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ and (b) clamped on the right interface $\boldsymbol{b}$ leading to $\boldsymbol{K} \in \mathbb{R}^{6\times6}$

If a mechanical body is without any support, Fig. 4.4 (a), $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ has the earlier mentioned n_ϕ=6 rigid body modes $\boldsymbol{\phi}_r$ and can freely move in space without any force $\boldsymbol{f}$=$\boldsymbol{0}$. Only by adding boundary conditions as, e.g., a displacement boundary condition, this system can be solved by removing those rigid body modes. If the mechanical body is, for instance, clamped on one side, Fig. 4.4 (b), then the interface stiffness matrix is $\boldsymbol{K} \in \mathbb{R}^{6\times6}$, while still possessing all information of the mechanical body. Thus, requirement (R2) combined with the symmetry requirement (R1) lead to n_k=21 independent stiffness entries $k_{i,j}$ of $\boldsymbol{K} \in \mathbb{R}^{6\times6}$. Since $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ can be recomputed from $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ by just taking the geometrical information of the rigid body modes $\boldsymbol{\phi}_r$ into account, also $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ possesses n_k=21 independent stiffness entries $k_{i,j}$. To establish the connection between the lower dimensional space $\mathbb{R}^6$ and $\mathbb{R}^{12}$, a linear map $\boldsymbol{\Phi}$ can be set up by solving (4.6) for specific $k_{i,j}$. Those $k_{i,j}$ are then the entries of the low-dimensional representation $\boldsymbol{\kappa}$, which is referred to as κ-representation in the following.

Since (4.6) is an underdetermined system of linear equations, there are several ways of parametrizing the linear map with 21 input values $k_{i,j}$, which does not necessarily need to be the entries of the clamped mechanical body of Fig. 4.4 (b). One possible representation is

$$\boldsymbol{\Phi}(\boldsymbol{\kappa}) = \boldsymbol{K}, \quad \boldsymbol{\Phi} : \mathbb{R}^6 \rightarrow \mathbb{R}^{12}, \tag{4.8}$$

$$\begin{aligned} \boldsymbol{\kappa} = \quad & [k_{1,1}, k_{2,1}, k_{3,1}, k_{4,1}, k_{5,1}, k_{6,1}, k_{2,2}, \ldots \\ & k_{2,3}, k_{2,4}, k_{2,5}, k_{3,3}, k_{3,4}, k_{3,6}, k_{4,4}, \ldots \\ & k_{4,5}, k_{4,6}, k_{5,5}, k_{5,6}, k_{6,6}, k_{11,11}, k_{12,12}], \end{aligned} \tag{4.9}$$

where $\boldsymbol{\kappa} \in \mathbb{R}^{1\times21}$. Thus n_k=21 stiffness entries determine the total deformation of the mechanical body with n_int=2 and n_dof=6. Note that unlike $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ of the clamped body in Fig. 4.4 (b), the diagonal entries $k_{11,11}$ and $k_{12,12}$ of interface $\boldsymbol{b}$ are taken instead of the non-diagonal entries $k_{2,6}$ and $k_{3,5}$ of $\boldsymbol{a}$. This is done because it is known from Section 2.1.1 that the diagonal entries of the positive semi-definite stiffness matrix $\boldsymbol{K}$ are strictly positive. Therefore, the lower bound on the diagonal entries is always known a priori, which facilitates subsequent design optimization for the stiffness entries.

The modeling process to connect the detailed design variables $\boldsymbol{x}$ of level (III) to the level (II) interface stiffness matrix $\boldsymbol{K}$ with respect to $\boldsymbol{a}$ and $\boldsymbol{b}$, see also Fig. 4.5, consists of three steps:

(1) Discretization of the design domain Ω of the mechanical body with three-dimensional brick elements $\boldsymbol{K}_e$ (see also Section 2.1.1),
(2) Static condensation of the design domain with respect to the left ($\boldsymbol{A}$) and right side ($\boldsymbol{B}$) of the domain using a Guyan reduction (see also Section 2.1.2),
(3) Kinematic condensation with respect to the n_dof=6 dimensional interfaces at $\boldsymbol{a}$ and $\boldsymbol{b}$ using a RBE2 formulation (see also Section 2.1.3).

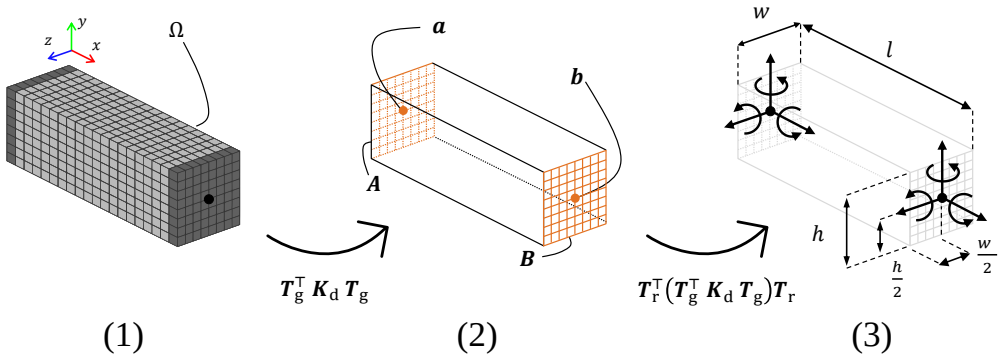


Figure 4.5: Modeling process of the interface stiffness matrix $\boldsymbol{K}$ consisting of (1) discretization, (2) static condensation, and (3) kinematic condensation

First (1), the geometrical design domain Ω of each structural element is discretized with n_{ele} three-dimensional brick elements $\boldsymbol{K}_e$. Within this work, the design domain Ω is realized with a rectangular domain of height h, width w, and length l. The numbering convention of the finite element mesh can be seen in Appendix A.1. The elastic behavior of the entire mechanical body is described by the detailed stiffness matrix $\boldsymbol{K}_{\text{d}}$

$$\boldsymbol{K}_{\text{d}} = \sum_{e=1}^{n_{\text{ele}}} \rho_e^p \boldsymbol{K}_e, \qquad 0 < \rho_e \leq 1, \tag{4.10}$$

while $\rho_e{=}x_j$ are the detailed design variables of the design domain Ω and p is a penalty factor utilized in SIMP-based topology optimization methods, see also Section 2.3.2. The design variables ρ_e scale each element $\boldsymbol{K}_e$ with respect to the given material properties, while the geometrical hull defined by h, w, and l remains constant. For $p{=}3$, this should lead to a 0−1-detailed design pattern were each finite element is either full material or void.

Then (2), a static condensation is performed with respect to the master degrees of freedom located on the left $\boldsymbol{A} \cup \boldsymbol{a}$ and right side $\boldsymbol{B} \cup \boldsymbol{b}$ of the design domain, Fig. 4.5 (2),

$$\boldsymbol{K}_{\text{g}} = \boldsymbol{T}_{\text{g}}^\top \boldsymbol{K}_{\text{d}} \boldsymbol{T}_{\text{g}}, \tag{4.11}$$

where $\boldsymbol{T}_{\text{g}}$ is the Guyan condensation matrix.

Finally (3), the remaining degrees of freedom on $\boldsymbol{A}$ and $\boldsymbol{B}$ are rigidly connected to the interfaces $\boldsymbol{a}$ and $\boldsymbol{b}$ using a RBE2 formulation of a rigid body element. The interface stiffness matrix $\boldsymbol{K}$ can be computed as

$$\boldsymbol{K} = \boldsymbol{K}_{\text{rg}} = \boldsymbol{T}_{\text{r}}^\top \boldsymbol{K}_{\text{g}} \boldsymbol{T}_{\text{r}} = \boldsymbol{T}_{\text{r}}^\top \boldsymbol{T}_{\text{g}}^\top \boldsymbol{K}_{\text{d}} \boldsymbol{T}_{\text{g}} \boldsymbol{T}_{\text{r}}, \tag{4.12}$$

with $\boldsymbol{T}_{\text{r}}$ being the second interface condensation matrix related to the rigid body elements on both sides, see also Appendix A.2.1.

Thus, the resulting structural element has two rigid interfaces on the left $\boldsymbol{A}$ and right side $\boldsymbol{B}$. The shown reduction process can be executed with no loss of information for the static case, meaning the results are equivalent to a body that only has a rigid left and right side, and no Guyan reduction (4.11). It should be noted that the reduction procedure to compute the interface stiffness matrix $\boldsymbol{K}$ can be carried out with different finite element types. For example, in Krischer & Zimmermann (2021), both beam elements and plane stress elements were used, where for beam elements with their rotational-based formulation, only a static Guyan condensation is needed but no rigid body elements.

The equation (4.12) establishes a connection between the detail level (III) design variables $\boldsymbol{x}$ and the component-performance level (II) quantity $\boldsymbol{K}$, which can be represented by the lower-dimensional $\boldsymbol{\kappa}$ vector of the linear mapping $\boldsymbol{\Phi}(\boldsymbol{\kappa})$. The computation of the mass m can be done directly from the detailed design variables $\boldsymbol{x}$. Later on, a connection between both level (II) quantities $\boldsymbol{y} = [m, \boldsymbol{K}]$ is established by the meta models $\hat{p}$ and $\hat{m}$, where the detail level (III) design variables $\boldsymbol{x}$ are incorporated implicitly by providing a sufficiently large sample dataset. The input of the meta models is then the lower-dimensional $\boldsymbol{\kappa}$ of $\boldsymbol{K} = \boldsymbol{\Phi}(\boldsymbol{\kappa})$, while the outputs are the second level (II) quantity m and the physical feasibility p

$$p \approx \hat{p}(\boldsymbol{\kappa}), \tag{4.13}$$

$$m \approx \hat{m}(\boldsymbol{\kappa}). \tag{4.14}$$

4.1.2 Symmetry conditions

To further reduce the number of dimensions of $\boldsymbol{\kappa} \in \mathbb{R}^{1\times 21}$, additional restrictions are imposed on the physically feasible detailed designs $\boldsymbol{x}$. First, it is assumed that the geometrical shape of the design domain Ω is invariant with respect to the x-axis, i.e., the dimension of the outer hull of the design domain Ω remains constant along the x-axis, as is the case, for example, for the rectangular design domain used, see Fig. 4.6. Thus, for topology optimization, only the design variables within the design domain can be changed. Second, a planar symmetry condition for the mechanical body is enforced with respect to the

(S1) $x-y$ plane and
(S2) $x-z$ plane.

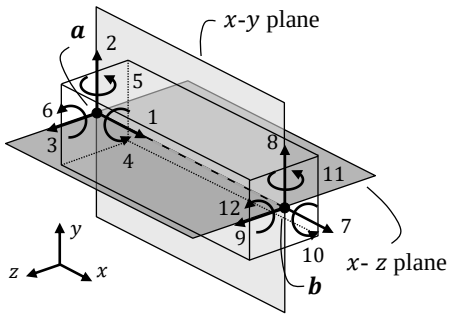


Figure 4.6: Exemplary invariant rectangular design domain with enforced plane symmetry on the x-y and x-z plane

The invariant design domain together with the symmetry conditions (S1) and (S2) lead to a beam-like deformation behavior of the structural element with respect to the interfaces $\boldsymbol{a}$ and $\boldsymbol{b}$. Following the idea of three-dimensional elastic beam theory, see, e.g., Andersen & Nielsen (2008), the interfaces now lie on the neutral axis of the mechanical body, meaning normal forces f_1 or f_7 only induce axial displacements with respect to d_1 and d_7. Thus,

$$k_{i,j} = 0, \quad \text{for } i = [1, 7] \text{ and } j \neq [1, 7]. \tag{4.15}$$

Also, $\boldsymbol{a}$ and $\boldsymbol{b}$ coincide with the principal axis of the bending moments f_5, f_{11} and f_6, f_{12} causing that only their associated shear loads f_3, f_9 and f_2, f_8, respectively, are coupled to these bending degrees of freedom. Hence, also all other stiffness entries $k_{i,j}$ are zero

$$k_{i,j} = 0, \quad \text{for } i = [2, 6, 8, 12] \text{ and } j \neq [2, 6, 8, 12], \tag{4.16}$$

$$k_{i,j} = 0, \quad \text{for } i = [3, 5, 9, 11] \text{ and } j \neq [3, 5, 9, 11]. \tag{4.17}$$

Finally, for double symmetric cross sections, as it holds for (S1) and (S2), also the neutral axis coincides with the shear center, meaning that torsional displacements d_4 and d_{10} are only caused by the torsional loads f_4 or f_{10}. Therefore,

$$k_{i,j} = 0, \quad \text{for } i = [4, 10] \text{ and } j \neq [4, 10], \tag{4.18}$$

are zero. With these assumptions, the interface stiffness matrix can approximately be determined by

$$\boldsymbol{K} = \begin{bmatrix} k_{1,1} & 0 & 0 & 0 & 0 & 0 & k_{1,7} & 0 & 0 & 0 & 0 & 0 \\ & k_{2,2} & 0 & 0 & 0 & k_{2,6} & 0 & k_{2,8} & 0 & 0 & 0 & k_{2,12} \\ & & k_{3,3} & 0 & k_{3,5} & 0 & 0 & 0 & k_{3,9} & 0 & k_{3,11} & 0 \\ & & & k_{4,4} & 0 & 0 & 0 & 0 & 0 & k_{4,10} & 0 & 0 \\ & & & & k_{5,5} & 0 & 0 & 0 & k_{5,9} & 0 & k_{5,11} & 0 \\ & & & & & k_{6,6} & 0 & k_{6,8} & 0 & 0 & 0 & k_{6,12} \\ & & & & & & k_{7,7} & 0 & 0 & 0 & 0 & 0 \\ & & & \text{sym} & & & & k_{8,8} & 0 & 0 & 0 & k_{8,12} \\ & & & & & & & & k_{9,9} & 0 & k_{9,11} & 0 \\ & & & & & & & & & k_{10,10} & 0 & 0 \\ & & & & & & & & & & k_{11,11} & 0 \\ & & & & & & & & & & & k_{12,12} \end{bmatrix}. \tag{4.19}$$

Incorporating (4.19) into (4.8) and (4.9) leads to a further reduced stiffness description

$$\boldsymbol{\Phi}_{\text{sym}}(\boldsymbol{\kappa}) = \quad \boldsymbol{K}, \tag{4.20}$$

$$\boldsymbol{\kappa}_{\text{sym}} = \quad [k_{1,1}, k_{2,2}, k_{3,3}, k_{4,4}, k_{5,5}, k_{6,6}, k_{11,11}, k_{12,12}\,], \tag{4.21}$$

where $\boldsymbol{\kappa}_{\text{sym}} \in \mathbb{R}^{1\times 8}$. Note that $\boldsymbol{\kappa}_{\text{sym}}$ only consist of diagonal entries of $\boldsymbol{K}$. In conclusion, n_{k}=8 diagonal stiffness entries are necessary to fully describe the deformation behavior $\boldsymbol{K} \in \mathbb{R}^{12\times 12}$ of the structural element with respect to $\boldsymbol{a}$ and $\boldsymbol{b}$ for an invariant design domain Ω and enforced symmetry planes $x{-}y$ and $x{-}z$. The explicit linear mapping $\boldsymbol{\Phi}_{\text{sym}}(\boldsymbol{\kappa}) = \boldsymbol{K}$ of (4.20) is given in the Appendix A.3.

4.1.3 Scaling

In contrast to e.g., shape optimization, topology optimization is performed on a predefined geometrical design domain Ω. On closer inspection, $\boldsymbol{K}$ and thus $\boldsymbol{\kappa}$ depend not only on the design variables $x_j{=}\rho_e$ related to the discretized domain, but also on the material parameters Young's modulus E and Poisson's ratio ν as well as the geometrical design domain Ω itself. The geometrical design domain Ω can then be determined, for example, by the parameters l, h, and w, see Fig 4.7 (a).

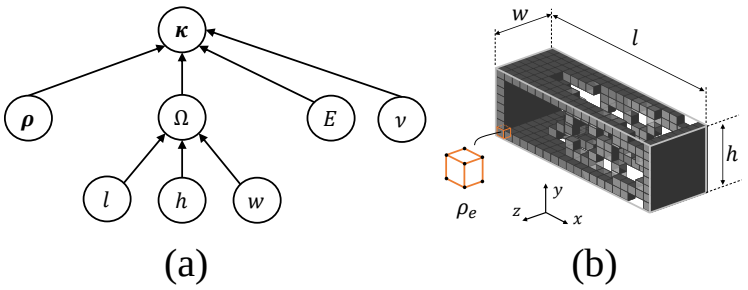


Figure 4.7: (a) Scaling dependencies between stiffness $\boldsymbol{\kappa}$ and the design variables $\boldsymbol{\rho}$, the parameters l, h, and w defining Ω, and the material parameters E and ν. (b) The discretized design domain Ω and one entry ρ_e of the numerical design variable vector $\boldsymbol{\rho}$

For a fixed discretization of the design domain Ω, see Fig 4.7 (b), establishing a link between same detailed designs $\boldsymbol{\rho}$ on different scaled $\Omega_{(i)}$ would be beneficial since the same meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ could be used for different geometrical dimensions. This supports the implementation of an offline database for the proposed decoupled optimization architecture (B).

Similitude theory is a branch of engineering that deals with establishing the necessary and sufficient conditions for similarity between a reference model and a scaled (up or down) model. It has been applied in various fields such as civil engineering, vibrations, and impact problems, and helps engineers to accurately predict the behavior of models (Coutinho et al., 2016; Casaburo et al., 2019). The main application of similitude methods is in the field of prototyping, but it can be also used for similarity analyses between computational models of different geometrical sizes. Therefore, the notion of scaling is introduced as a means of changing the geometrical dimensions of a structure in order to analyze its response to external influences. For a reference stiffness $\boldsymbol{\kappa}_{\text{ref}}$ based on $\boldsymbol{\rho}$, the scaling of a rectangular design domain Ω defined by l, h, and w can be achieved by multiplying it with a scaling factor $\boldsymbol{\alpha}$. It is assumed that the relation between the new stiffness $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}_{\text{ref}}$ for same $\boldsymbol{\rho}$ can be defined as

$$\boldsymbol{\kappa}_{\text{ref}} = \boldsymbol{\lambda}(\boldsymbol{\alpha}) \circ \boldsymbol{\kappa}, \tag{4.22}$$

where $\boldsymbol{\lambda}$ is a scaling parameter and $\circ$ is the Hadamard product, i.e., an element-wise multiplication operator of the respective vector entries.

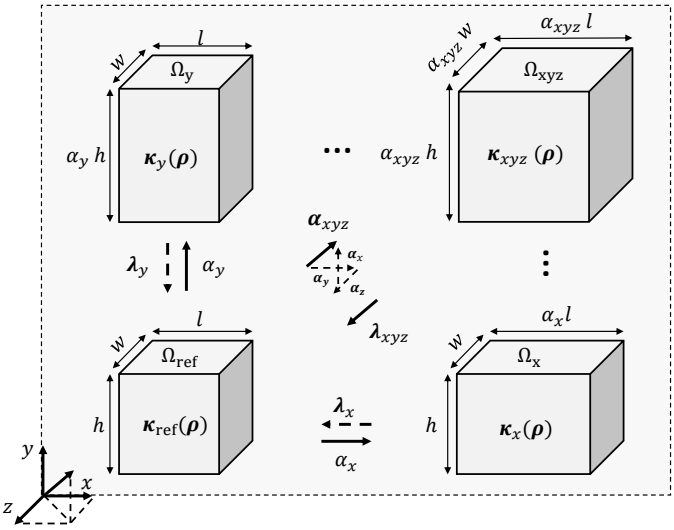


Figure 4.8: Similarity mapping between different $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}_{ref}$ for same detailed designs $\boldsymbol{\rho}$ and different geometrical design domains $\Omega_{(i)}$

Fig. 4.8 illustrates the relationship between a reference model and scaled models in detail. Starting from a given detailed design $\boldsymbol{\rho}$, the model is scaled with the factor $\boldsymbol{\alpha}$, while the scaling parameter $\boldsymbol{\lambda}$ specifies the scaling laws between similar components. Note that scaling does not work for every change in the design domain and also depends on the underlying mechanical model. If a suitable scaling parameter $\boldsymbol{\lambda}$ can be found, the stiffness $\boldsymbol{\kappa}$ can can be mapped back onto $\boldsymbol{\kappa}_{\text{ref}}$ using (4.22).

For a three-dimensional body Ω without interfaces $\boldsymbol{a}$ and $\boldsymbol{b}$, that is discretized with volumetric brick elements, the detailed design $\boldsymbol{\rho}$ determines its elastic behavior. For a constant scaling $\boldsymbol{\alpha}_{xyz}$ of the domain Ω in all coordinate directions, a scaling parameter $\boldsymbol{\lambda}_{xyz}$ can be directly derived from the local formulation $\boldsymbol{K}_e$ of the element stiffness matrix as

$$\boldsymbol{\lambda}_{xyz} = \frac{1}{\boldsymbol{\alpha}_{xyz}}. \tag{4.23}$$

In general, a scaling with $\boldsymbol{\alpha}_{xyz}$ affects the coordinates of the geometrical design domain Ω of the structural element. If the reduction procedure to compute the κ-representation (4.21) is considered, it can be observed that the distance vector $\Delta\boldsymbol{x}$ between master and slave nodes of the geometrical constraint (2.12) of the rigid body element is altered

$$\boldsymbol{\alpha}_{xyz} \circ [\boldsymbol{x}_s - \boldsymbol{x}_m], \tag{4.24}$$

and hence, the geometrical constraint (2.12) is also affected by taking the cross product of (4.24) with the rotational degrees of freedom $\boldsymbol{\varphi}_m$

$$\begin{bmatrix} \boldsymbol{u}_s \\ \boldsymbol{\varphi}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_m + \boldsymbol{\varphi}_m \times \boldsymbol{\alpha}_{xyz} \circ [\boldsymbol{x}_s - \boldsymbol{x}_m] \\ \boldsymbol{\varphi}_m \end{bmatrix}. \tag{4.25}$$

The resulting interface stiffness matrix $\boldsymbol{K}$ is calculated multiplying the interface condensation matrix $\boldsymbol{T}_\mathrm{r}$ on both sides of $\boldsymbol{K}_\mathrm{g}$

$$\boldsymbol{K} = \boldsymbol{T}_\mathrm{r}^\top \boldsymbol{K}_\mathrm{g} \boldsymbol{T}_\mathrm{r},$$

see also equation (4.12), and $\boldsymbol{T}_\mathrm{r}$ is computed using the geometrical constraint (4.25) containing $\boldsymbol{\alpha}_{xyz}$.

Therefore, the scaling parameter $\boldsymbol{\lambda}_{xyz}$ of the κ-representation (4.21) for constant scaling $\boldsymbol{\alpha}_{xyz}$ in all coordinate directions is affected by (1) the discretization scaling parameter of (4.23) and (2) the interface condensation matrices $\boldsymbol{T}_\mathrm{r}$

$$\boldsymbol{\lambda}_{xyz} = \left[\frac{1}{\alpha}, \frac{1}{\alpha}, \frac{1}{\alpha}, \frac{1}{\alpha^3}, \frac{1}{\alpha^3}, \frac{1}{\alpha^3}, \frac{1}{\alpha^3}, \frac{1}{\alpha^3} \right]_{xyz}, \quad \boldsymbol{\lambda}_{xyz} \in \mathbb{R}^{1\times 8}. \tag{4.26}$$

Note that the translational degrees of freedom $\boldsymbol{u}$ are solely affected by (4.23), while the rotational ones, $\boldsymbol{\varphi}$, have a cubic relationship $(\,\cdot\,)^3$ due to the geometrical constraint (4.25) considered in $\boldsymbol{T}_\mathrm{r}$. For scaling with respect to a single coordinate axis, no matching parameters $\boldsymbol{\lambda}_x$, $\boldsymbol{\lambda}_y$ or $\boldsymbol{\lambda}_z$ could be identified.

For arbitrarily geometrical scaling, the meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ can be extended to consider the geometrical scaling factor $\boldsymbol{\alpha}$ explicitly. Due to the known scaling law (4.26) of $\boldsymbol{\lambda}_{xyz}$ for constant scaling, only n_α=2 more explicit α are needed for arbitrary changes in all coordinate directions, e.g.,

$$\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha}) = \hat{p}(\boldsymbol{\kappa}, \alpha_x, \alpha_y), \tag{4.27}$$

$$\hat{m}(\boldsymbol{\kappa}, \boldsymbol{\alpha}) = \hat{m}(\boldsymbol{\kappa}, \alpha_x, \alpha_y). \tag{4.28}$$

In Fig. 4.9 an exemplary scaling is shown. Direct scaling in the z-direction (α_z) is replaced by first scaling the reference design domain Ω_ref with a constant scale $\boldsymbol{\alpha}_{xyz}$ in all coordinate directions (1). Then, α_x (2) and α_y (3) are used to adjust the dimensions in the other two dimensions to realize the desired scaling in the z-direction, while the dimensions in the x- and y-directions correspond to the original reference design domain Ω_ref.

Apart from scaling the geometrical dimensions of the design domain Ω, also a constitutive scaling with respect to different materials would enhance the application area of the offline database. As long as the Poisson's ratio is the same $\nu_\mathrm{ref} = \nu_{(i)}$, $\boldsymbol{\kappa}$ can be scaled right away applying the linear relationship

$$\lambda_c = \frac{E_\mathrm{ref}}{E}. \tag{4.29}$$

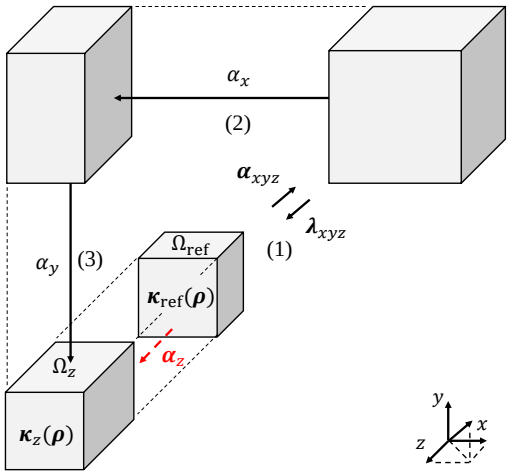


Figure 4.9: Scaling of the design domain Ω in z-direction without α_z, but α_x, α_x, and α_{xyz}

The scaled κ-representations for each component i can then be used as a low-dimensional representation of the interface stiffness matrices $\boldsymbol{K}_{(i)} \in \mathbb{R}^{12\times12}$ and are, together with the scaling factors $\boldsymbol{\alpha}$, the input parameters for the meta models $\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ and $\hat{m}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ of the system optimization in the following section.

4.2 Decoupled optimization architecture

4.2.1 System optimization

For available $\hat{p}$ and $\hat{m}$ in the offline database, the system optimization decouples the given design problem with respect to the physical components, i.e., it performs an object-based partitioning. The system optimization problem is a variation of the initial monolithic optimization (A) of (1.3). However, in contrast to the monolithic optimization, the system optimization uses $\boldsymbol{\kappa}$ (II) instead of $\boldsymbol{x}$ (III) as design variables, see Fig 4.10 (a)-(b).

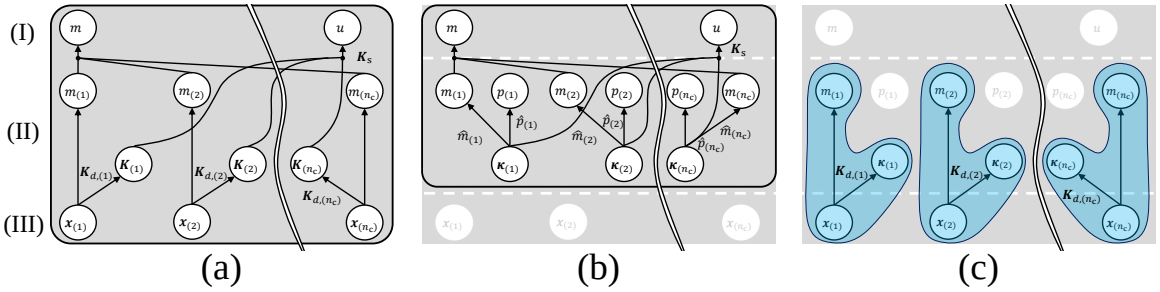


Figure 4.10: Dependencies between all relevant quantities on the system level (I), component-performance level (II), and component-detail level (III) in (a) monolithic optimization and the proposed approach consisting of (b) system and (c) component optimization

This means, that the system optimization is carried out without knowing the component details. To ensure feasibility and optimality, a surrogated-based system optimization using the meta models $\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ and $\hat{m}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ is adopted. The system optimization problem reads

$$\begin{aligned} & \min_{\boldsymbol{\kappa}_{(i)}} \sum_{i=1}^{n_\mathrm{c}} \hat{m}(\boldsymbol{\kappa}_{(i)}, \boldsymbol{\alpha}_{(i)}), \\ \text{s. t.:} \quad & -\hat{p}\left(\boldsymbol{\kappa}_{(i)}, \boldsymbol{\alpha}_{(i)}\right) + 1 = 0, \\ & u_c\left(\boldsymbol{\kappa}_{(i)}\right) - u_{\max} \leq 0, \qquad \text{for } c{=}1, ..., n_\mathrm{p}, \\ & \boldsymbol{\kappa}_\mathrm{lb} \leq \boldsymbol{\kappa}_{(i)} \leq \boldsymbol{\kappa}_\mathrm{ub}, \qquad \text{for } i = 1, ..., n_\mathrm{c}. \end{aligned} \tag{4.30}$$

The mass estimator $\hat{m}(\boldsymbol{\kappa}_{(i)}, \boldsymbol{\alpha}_{(i)})$ enables mass estimates with respect to $\boldsymbol{\kappa}_{(i)}$ for a specified design domain $\Omega_{(i)}$. The feasibility estimator $\hat{p}(\boldsymbol{\kappa}_{(i)}, \boldsymbol{\alpha}_{(i)})$ indicates whether the component's stiffness $\boldsymbol{\kappa}_{(i)}$ corresponds to an actual physical design $\boldsymbol{x}_{(i)}$, hence $\hat{p}(\boldsymbol{\kappa}_{(i)}, \boldsymbol{\alpha}_{(i)})$=1. In comparison to the monolithic optimization of (1.3), the system optimization problem is extended with constraints on the physical feasibility estimator $\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ for each component i ensuring physical feasibility. Note that for computational reasons usually the actual mass $m_{(i)}$ is not computed. Instead, the volume fraction $v_{(i)}{=}\frac{100\%}{V_\mathrm{ref}} \frac{m_{(i)}}{\rho}$ for a design domain $\Omega_{(i)}$ with reference volume V_ref and material of density ρ is utilized. For systems consisting of components of same material but with different geometrical dimensions, weighting $w_{(i)}$ is necessary, $v = \sum_{i=1}^{n_\mathrm{c}} w_{(i)} v_{(i)}$, because the volume fraction always remains between $0{\leq}v_{(i)}{\leq}100\%$, and hence is independent of the actual dimensions of the design domain $\Omega_{(i)}$.

The formulation with respect to $\boldsymbol{\kappa}$ has two advantages: First, the system optimization does not need to compute the high dimensional detailed stiffness matrices $\boldsymbol{K}_{d,(i)}$, but only the low-dimensional interface stiffness matrices $\boldsymbol{K}_{(i)}$. This reduces the computational cost of solving the bottom-up mapping for computing the displacements $\boldsymbol{d}_s$ on the system level needed for the requirement on the translational displacements $u = ||\boldsymbol{u}_\mathrm{ee}||_2$. Second, the optimization driver itself has usually fewer design variables compared to monolithic optimization. When using, e.g., topology optimization, the number of design variables equals the number of elements in the model, whereas for the system optimization only

the reduced stiffness vectors $\boldsymbol{\kappa}_{(i)}$ needs to be optimized. However, due to the binary classification characteristic of $\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ and the possibly non-convex design domain spanned by $\hat{p}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$ and $\hat{m}(\boldsymbol{\kappa}, \boldsymbol{\alpha})$, a gradient-free and global-search algorithm, namely a particle swarm optimization (PSO) is utilized to ensure global convergence, see also Section 2.3.1.

In order to establish the bottom-up mapping for the system optimization, a given multi-component system needs to be assembled, see also Fig. 4.11. Therefore, each interface stiffness matrix $\boldsymbol{K}_{(i)}$ related to $\boldsymbol{a}_{(i)}$ and $\boldsymbol{b}_{(i)}$ is first rigidly connected to the joint positions $\boldsymbol{p}_{(i-1)}$ and $\boldsymbol{p}_{(i)}$, i.e.,

$$\boldsymbol{K}_{\mathrm{p},(i)} = \boldsymbol{T}_{\mathrm{p},(i-1,i)}^{\top} \boldsymbol{K}_{(i)} \boldsymbol{T}_{\mathrm{p},(i-1,i)}, \tag{4.31}$$

utilizing a RBE2 formulation via the joint condensation matrix $\boldsymbol{T}_{\mathrm{p},(i-1,i)}$, see Appendix A.2.2.

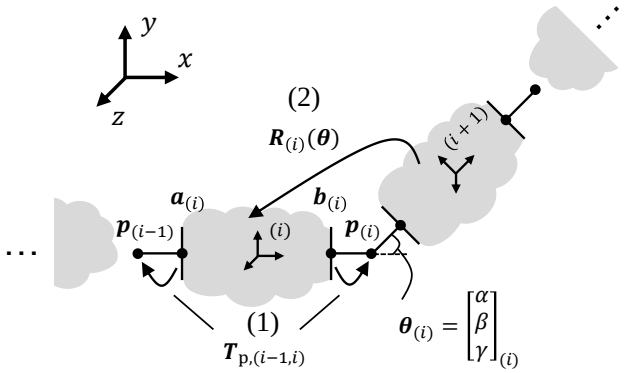


Figure 4.11: Assembly process consisting of (1) the condensation matrix $\boldsymbol{T}_{p,(i-1,i)}$ that connects $\boldsymbol{a}_{(i)}$ and $\boldsymbol{b}_{(i)}$ via rigid body elements to $\boldsymbol{p}_{(i-1)}$ and $\boldsymbol{p}_{(i)}$, and (2) the rotation matrix $\boldsymbol{R}_{(i)}$

Afterwards, the joint stiffness matrix $\boldsymbol{K}_{\mathrm{p},(i)}$ is transformed into the global coordinate system

$${}^{(0)}\boldsymbol{K}_{\mathrm{p},(i)} = \boldsymbol{R}_{(0)}^{\top} \ldots \boldsymbol{R}_{(i-1)}^{\top} \boldsymbol{K}_{\mathrm{p},(i)} \boldsymbol{R}_{(i-1)} \ldots \boldsymbol{R}_{(0)}, \tag{4.32}$$

applying the rotation matrices $\boldsymbol{R}_{(j)}$ with the respective rotations $\boldsymbol{\theta}_{(j)}$ at the joint positions $\boldsymbol{p}_{(j)}$ for $j = 0, ..., i-1$.

Finally, the global multi-component system stiffness matrix can be assembled

$$\boldsymbol{K}_{\mathrm{s}} = \bigwedge_{i=1}^{n_{\mathrm{c}}} {}^{(0)}\boldsymbol{K}_{\mathrm{p},(i)}, \tag{4.33}$$

and the system displacement is computed as

$$\boldsymbol{d}_{\mathrm{s}} = \boldsymbol{K}_{\mathrm{s}}^{-1} \boldsymbol{f}_{\mathrm{s}}, \tag{4.34}$$

with $\boldsymbol{u}_{\mathrm{ee}} \in \boldsymbol{d}_{\mathrm{s}}$ as the translational displacements of the end effector. The design variables for the system optimization of a n_{c}-component system are

$$\boldsymbol{\kappa} = [\boldsymbol{\kappa}_{(1)}, ..., \boldsymbol{\kappa}_{(n_{\mathrm{c}})}]. \tag{4.35}$$

After the system optimization has been carried out, each optimized $\boldsymbol{\kappa}_{(i)}^{*}$ can be now taken as a target stiffness for the subsequent n_{c} decoupled component optimization problems

$$\boldsymbol{\kappa}_{(i)}^{t} = \boldsymbol{\kappa}_{(i)}^{*}. \tag{4.36}$$

4.2.2 Component optimization

The component optimization seeks the optimal design variables $\boldsymbol{x}_{(i)}$ (III) in the geometrical design domain Ω with minimum mass $m_{(i)}$ for a given target interface stiffness $\boldsymbol{\kappa}^t_{(i)}$ (II), see Fig 4.10 (c).

The component optimization problem statement reads

$$\begin{aligned} &\min_{\boldsymbol{x}_{(i)}} m_{(i)}(\boldsymbol{x}_{(i)}), \\ \text{s. t.:} \quad &\|\boldsymbol{\kappa}_{(i)}(\boldsymbol{x}_{(i)}) - \boldsymbol{\kappa}^t_{(i)}\| \leq \epsilon, \\ &\boldsymbol{x}_{\text{lb}} \leq \boldsymbol{x}_{(i)} \leq \boldsymbol{x}_{\text{ub}}, \end{aligned} \tag{4.37}$$

where $\boldsymbol{x}_{\text{lb}}$ and $\boldsymbol{x}_{\text{ub}}$ are the lower and upper bounds on the design variables $\boldsymbol{x}_{(i)}$, $\boldsymbol{\kappa}_{(i)}$ is the low-dimensional representation of the interface stiffness matrix $\boldsymbol{\Phi}(\boldsymbol{\kappa}_{(i)})=\boldsymbol{K}_{(i)}$ associated with $\boldsymbol{x}_{(i)}$ and ϵ is a small positive value. The masses $m_{(i)}$ are again processed as volume fractions $v_{(i)}$, while now no estimator $\hat{m}$ is utilized, but the actual volume fraction is computed from the detailed design variables $\boldsymbol{x}_{(i)}$. The optimization problems (4.37) can be solved in parallel and independently of each other due to the decoupling of the system optimization (4.30). The system stiffness measured by the total displacement u is assumed to satisfy the requirement $u \leq u_{\max}$ as long as the component optimizations are all feasible.

For a given design domain Ω, discretized with three-dimensional brick elements $\boldsymbol{K}_e$, a three-dimensional topology optimization based on the SIMP method can be applied as already introduced in Section 4.1.1. The detailed design variables x_j then correspond to the element densities ρ_e and scale the element stiffness matrix $\boldsymbol{K}_e$ and also determine the volume fraction v of the given design domain Ω

$$\rho_e^p \boldsymbol{K}_e, \qquad x_j = \rho_e, \qquad p = 3, \tag{4.38}$$

$$v = \frac{100\%}{n_{\text{ele}}} \sum_{e=1}^{n_{\text{ele}}} \rho_e, \quad 0 < \rho_e \leq 1. \tag{4.39}$$

The penalty factor is hereby set to p=3, the sensitivity filter radius in the dimensionless parameter space for finite elements of side length l_e=w_e=h_e=1 is r=$\sqrt{2}$, and the element volume V_e is the same for all elements.

The MMA developed by Svanberg (1987) can be utilized to compute the solution of each component optimization problem. Since the MMA is a gradient-based optimization algorithm, it needs the derivatives of the interface stiffness matrix $\boldsymbol{K}$=$\boldsymbol{K}_{\text{rg}}$,

$$\frac{\partial \boldsymbol{K}_{\text{rg}}}{\partial \rho_e} = \boldsymbol{T}_{\text{r}}^\top \left(\boldsymbol{T}_{\text{g}}^\top \frac{\partial \boldsymbol{K}_{\text{d}}}{\partial \rho_e} \boldsymbol{T}_{\text{g}} \right) \boldsymbol{T}_{\text{r}}, \tag{4.40}$$

$$\frac{\partial \boldsymbol{K}_{\text{d}}}{\partial \rho_e} = p \rho_e^{p-1} \boldsymbol{K}_e, \tag{4.41}$$

as well as the volume fraction gradients

$$\frac{\partial v}{\partial \rho_e} = \frac{100\%}{n_{\text{ele}}}, \tag{4.42}$$

that are constant and relate to a 100% filled reference unit volume.

The stiffness constraint $g(\boldsymbol{x}_{(i)}) = \|\boldsymbol{\kappa}_{(i)}(\boldsymbol{x}_{(i)}) - \boldsymbol{\kappa}^t_{(i)}\| \leq \epsilon$ of (4.37) is reformulated for numerical processing as

$$-\epsilon \leq \left[\frac{\kappa_j - \kappa_j^t}{\kappa_{\mathrm{ub},j} - \kappa_{\mathrm{lb},j}} \right] \leq \epsilon, \quad \text{for } j = 1, ..., n_{\mathrm{k}}, \tag{4.43}$$

where κ_{ub} and κ_{lb} are the upper and lower bounds of the stiffness entries, respectively, and normalize the different stiffness units of $\boldsymbol{\kappa}$. This normalization is of utmost importance since constraints of the MMA should be in a range of $1 \leq g(\boldsymbol{x}) \leq 100$ (Svanberg, 2007). Experience here shows a high sensitivity to violations of this recommendation in terms of the MMA convergence behavior.

Since the component optimization (4.37) is a newly developed topology optimization formulation, it is further recommended to check the reliability of the formulation for new discretized design domains Ω that are also not covered by similitude theory. For this purpose, a three step verification procedure is recommended before a multi-component system is optimized, see Fig. 4.12:

(1) the classical minimum compliance optimization (2.57) can be used to compute several reference stiffnesses $\boldsymbol{\kappa}_0$ for the resulting $\boldsymbol{x}_0$ for a given volume fraction v_0 and force $\boldsymbol{f}$,
(2) $\boldsymbol{\kappa}_0$ is then used as a target value for the component optimization (4.37) which computes the detailed design $\boldsymbol{x}$,
(3) the resulting $\boldsymbol{x}_0$ of the minimum compliance optimization (2.57) is compared to $\boldsymbol{x}$ of the component optimization (4.37). If $\boldsymbol{x}_0 \approx \boldsymbol{x}$, one can proceed with the Informed Decomposition approach (true). If $\boldsymbol{x}_0 \neq \boldsymbol{x}$ (false), either the heuristic parameters of the MMA itself, see Svanberg (2007), or general optimization settings, such as start vector, maximum number of iterations, or numerical processing of the constraint (4.43) can be changed.

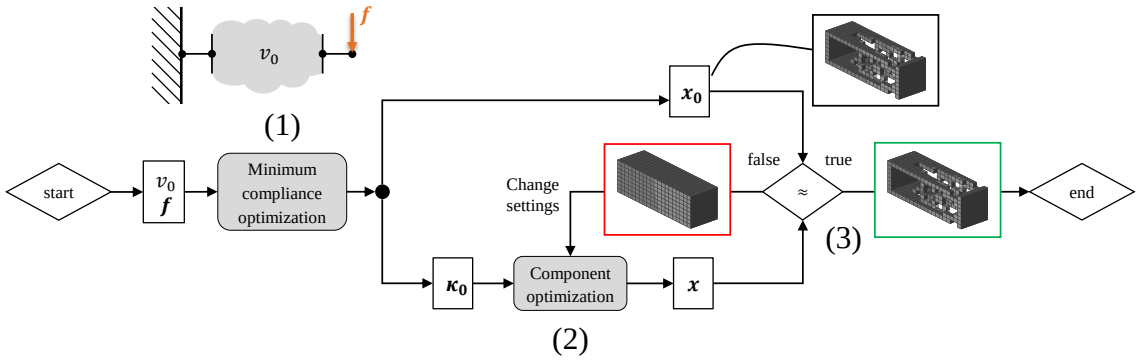


Figure 4.12: Verification procedure to ensure the convergence behavior of the component optimization (4.37) by comparing its results $\boldsymbol{x}$ to the results $\boldsymbol{x}_0$ of a classical minimum compliance problem (2.57) for a given $\boldsymbol{\kappa}_0$

4.3 Offline database

4.3.1 Setup

The organization of the offline database with meta models $\hat{p}_{(i)}$ and $\hat{m}_{(i)}$ for respective interface degrees of freedom n_{dof} and design domain Ω, material $[E, \nu, \rho]$, and discretization n_{ele} can be seen in Table 4.1.

Table 4.1: Organization of the offline database containing the trained meta models $\hat{p}_{(i)}$ and $\hat{m}_{(i)}$

n_{dof}	2	3	6
and meta models	$[\hat{p}, \hat{m}]_{(i)}$	$[\hat{p}, \hat{m}]_{(i)}$	$[\hat{p}, \hat{m}]_{(i)}$
with	Geometrical design domain $\Omega_{(i)}$		
	Material data $[E, \nu, \rho]_{(i)}$		
	Discretization $n_{\text{ele},(i)} = [n_{\text{ele},x}, n_{\text{ele},y}, n_{\text{ele},z}]_{(i)}$		

If no appropriate feasibility estimator $\hat{p}$ and mass estimator $\hat{m}$ can be found in the offline database, new meta models need to be established, see Fig. 4.1 (b). To provide the necessary training data $\boldsymbol{y}_A$, for $A = 1, ..., N$, first the input space needs to be sampled

$$\boldsymbol{y}_A = [\boldsymbol{\kappa}_A, \boldsymbol{\alpha}_A], \quad \boldsymbol{y}_A \in \mathbb{R}^{1 \times n_{\text{y}}}, \tag{4.44}$$

for n_{y}=($n_{\text{k}} + n_{\alpha}$) and within the bounds $[\boldsymbol{y}_{\text{lb}}, \boldsymbol{y}_{\text{ub}}]$. The sample output vector contains information about physical feasibility p and the mass m

$$\boldsymbol{z}_A = [p_A, m_A], \quad \boldsymbol{z}_A \in \mathbb{R}^{1 \times n_z}, \tag{4.45}$$

while m_A is again processed as v_A. In order to compute the sample output data $\boldsymbol{z}_A$, the component optimization of (4.37) is utilized. It determines whether a corresponding detailed design $\boldsymbol{x}$ with minimum mass m exists for the given $[\boldsymbol{\kappa}, \boldsymbol{\alpha}]$. If the component optimization converges, the feasibility flag is set to p_A=1, otherwise p_A=-1. The sample data consists then of the input and output data

$$[\boldsymbol{Y}, \boldsymbol{Z}], \quad \in \mathbb{R}^{N \times (n_y + n_z)}. \tag{4.46}$$

The sample data can then be used to train new feasibility estimator $\hat{p}$ and mass estimator $\hat{m}$ for the offline database. The feasibility estimator $\hat{p}$ is evaluated in terms of the false positive rate, FPR, true positive rate, TPR, and the accuracy, ACC, of Section 2.2.2, while the mass estimator $\hat{m}$ is evaluated in terms of the MSE and the R^2-value of Section 2.2.1.

4.3.2 Active-learning undersampling strategy

In general, not all combinations of stiffness entries can be physically realized. Physically feasible stiffness matrices $\boldsymbol{K}$ and thus $\boldsymbol{\kappa}$ must satisfy the requirements (R1-R4) of Section 4.1.1. Therefore, any randomly sampled dataset $[\boldsymbol{Y}, \boldsymbol{Z}]$ is assumed to contain many more infeasible than feasible data points. This is not only disadvantageous for the training process of the classifier, but also makes the process less efficient, because infeasible data points can only be used for the feasibility estimator $\hat{p}$, but not for the mass estimator $\hat{m}$. This is particularly problematic because the component optimizations are performed for each point at the high-dimensional component-detail level (III), making the computation very expensive. One way to deal with imbalanced training data is informed undersampling strategies. SVMs are particularly well suited for this type of sampling because of the mathematical definition of the separating hyperplane. Based on a SVM, a two-phase active-learning undersampling strategy was

proposed in Krischer et al. (2022), see also Fig. 4.13 (a), consisting of

(i) classification sampling and
(ii) regression sampling.

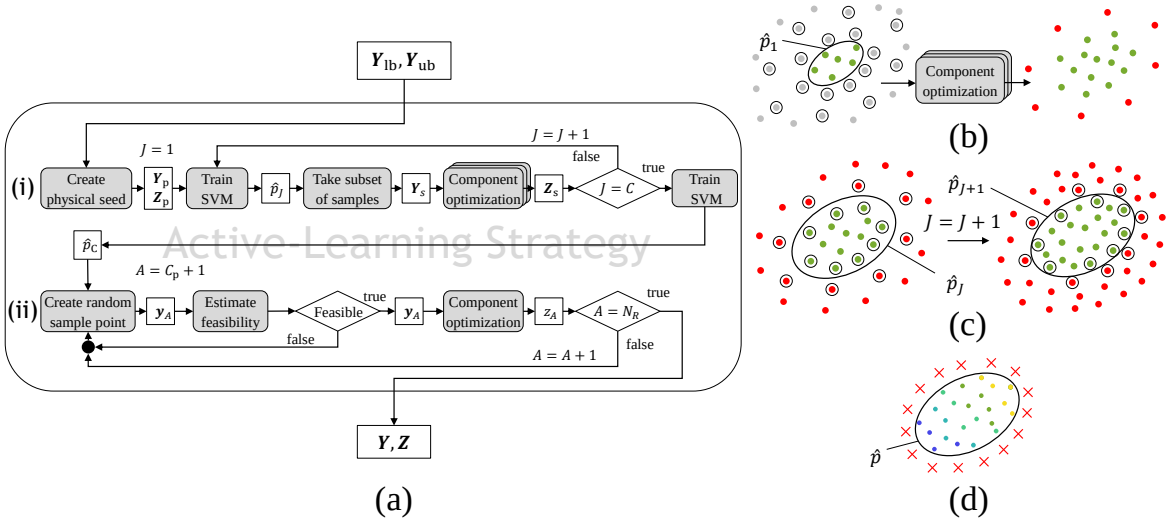


Figure 4.13: Active-learning undersampling strategy: (a) sampling process, (b) physical seed $\boldsymbol{Y}_p$ of iteration J=1 and evaluation of each sample point via component optimization, (c) decreasing distance of selected sample points to feasibility boundary for increasing number of iterations J, and (d) sample data for training mass estimator. Green, red, and gray dots show feasible, infeasible and ignored sample points, respectively, circles indicate selection, and red crosses mark infeasible designs that are ignored

Classification sampling (i) approximates the hyperplane between physically feasible and infeasible designs by selecting sample points according to temporally trained feasibility estimators $\hat{p}_J$. In each iteration J, a new SVM classifier $\hat{p}_J$ is trained on a growing set of training data to improve the selection quality of sample points $\boldsymbol{Y} \in \mathbb{R}^{N_C \times n_y}$ incrementally.

For the first iteration J=1, see also Fig. 4.13 (b), a physical seed $\boldsymbol{Y}_p$ is generated by randomly selecting the design variables of the detail level $\boldsymbol{x}_p$ and computing the respective $\boldsymbol{\kappa}_p$-values afterwards. Since the $\boldsymbol{\kappa}_p$ are computed from existing detailed designs $\boldsymbol{x}_p$, all sample points of the first iteration are physically feasible. Therefore, a one-class SVM is trained, based on the physically feasible seed to select the next input samples. A random set of samples $\boldsymbol{Y}_t \in \mathbb{R}^{N_t \times n_y}$ is created (gray dots) and evaluated by the corresponding estimator $\hat{p}_1$. Then, a subset of sample points $\boldsymbol{Y}_s \in \mathbb{R}^{N_s \times n_y}$ is added to the dataset (black circled points) that have the smallest distance from the estimated hyperplane, with $N_t >> N_s$. For each sample point the component optimization (4.37) is utilized to determine the respective outputs $z_A = [p_A, m_A]$.

For all following iterations J>1, a binary SVM is utilized (physically feasible/infeasible), see Fig. 4.13 (c). At each iteration, the sample size N_t of the temporary samples $\boldsymbol{Y}_t$ is increased, while the sample size N_s of the subset $\boldsymbol{Y}_s$ is kept constant. Since the ratio N_s/N_t decreases at each iteration, it is likely that the selected subset of sample points $\boldsymbol{Y}_s$ of iteration $J+1$ is closer to the trained hyperplane than the sample points of iteration J. After a given number of iterations C, a final classifier $\hat{p}_C$ is trained on the entire sample data and phase (i) of the proposed active-learning strategy is completed.

Next, regression sampling (ii) is initiated, where N_R regression samples are selected within the feasible region of the classifier $\hat{p}$, as shown in Fig. 4.13 (d). Note that all feasible points C_p from the previous phase (i) can already be added to the set of sample points. Each remaining sample point $\boldsymbol{y}_A$ is randomly generated within the boundaries of the input space and evaluated by the classifier $\hat{p}$. If the sample

point is found to be infeasible, the design is ignored, and a new point is generated and evaluated. If it is feasible, the expensive component optimization of (4.37) is performed to compute the corresponding mass m_A and re-evaluate the feasibility p_A. When the predefined number of sample points N_R is reached, phase (ii) is completed and the sample data can be used to train the final meta models $\hat{m}$ and $\hat{p}$ with respect to the whole dataset $[\boldsymbol{Y}, \boldsymbol{Z}]$, as shown in Fig. 4.1 (b). For simplicity, one can also take the estimator $\hat{p}_C$=$\hat{p}$ of phase (i) right away as the final feasibility estimator. All important parameters of the active-learning strategy are also summarized in Table 4.2.

Table 4.2: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase

(i) Classification sampling: $N_J = N_\mathrm{p}, \ldots, N_\mathrm{C}$ for $J = 1, \ldots, C$			
Physical seed $\boldsymbol{Y}_\mathrm{p}$	Classification samples $\boldsymbol{Y}_\mathrm{C}$	Temporary samples $\boldsymbol{Y}_\mathrm{t}$	Subset of samples $\boldsymbol{Y}_\mathrm{s}$
$\boldsymbol{Y}_\mathrm{p} \in \mathbb{R}^{N_\mathrm{p} \times n_\mathrm{y}}$	$\boldsymbol{Y}_\mathrm{C} \in \mathbb{R}^{N_\mathrm{C} \times n_\mathrm{y}}$	$\boldsymbol{Y}_\mathrm{t} \in \mathbb{R}^{N_\mathrm{t} \times n_\mathrm{y}}$, for $N_\mathrm{t} = J^p \frac{(N_\mathrm{ub,t} - N_\mathrm{lb,t})}{C^p} + N_\mathrm{lb,t}$	$\boldsymbol{Y}_\mathrm{s} \in \mathbb{R}^{N_\mathrm{s} \times n_\mathrm{y}}$
(ii) Regression sampling: for $A = C_\mathrm{p} + 1, \ldots, N_\mathrm{R}$			
Feasible classification samples $\boldsymbol{Y}_C = 1$		Regression samples $\boldsymbol{Y}_R$	
$\boldsymbol{Y}_C \in \mathbb{R}^{C_\mathrm{p} \times n_\mathrm{y}}$		$\boldsymbol{Y}_R \in \mathbb{R}^{N_\mathrm{R} \times n_\mathrm{y}}$	

The overall success of the active-learning undersampling strategy depends mainly on the component optimization (4.37), as it provides us with the crucial information about the feasibility p and the mass m. A mislabeled sample point can drastically degrade the results of the feasibility estimator $\hat{p}$, while a non-convergent optimization produces incorrect mass values m that affect $\hat{m}$. If the verification of Section 4.2.2 was successful, a reliable optimization is assumed.

In addition, it is important to assess the extent to which the input design space of $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$ has been sampled. This can be done by using some precomputed specific designs that are known to be located at outer points of the physically feasible design space. Since the κ_sym-representation (4.21) consists only of diagonal values, it is known that a completely filled design domain Ω represents the maximum value for a given Ω, while the minimum value is obtained for a void design domain Ω. In addition, the results of the component verification procedure of Fig. 4.12 can be also used to assess the design space size. Together, a non-precise but rough estimate of the design space can be realized, by either visually examining the sampled design space, see Fig. 4.14, or giving the produced designs $\boldsymbol{\kappa}$ as an input to the trained feasibility estimator $\hat{p}$.

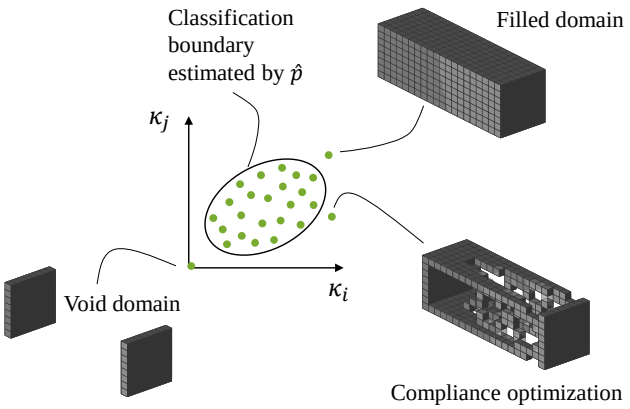


Figure 4.14: A completely filled and void design domain can be used to determine the maximum and minimum value of the design space. Additionally, results of a compliance optimization can be utilized to evaluate intermediate design points

4.4 Validation methodology

In order to validate the developed Informed Decomposition (B), two design problem classes are introduced:

(P1) the validity of the proposed approach is investigated by taking the physical feasibility p and optimality with respect to mass m into account, and
(P2) an investigation towards practical application is carried out.

For the two design problem classes (P1) and (P2), with n_c=2 and n_c=4 components, respectively, the complexity slowly increases by increasing the number of degrees of freedom n_{dof} and load cases n_p for each design problem, see Table 4.3.

First, in Chapter 5, the design problem class (P1) is studied. For this purpose, a two-component system is optimized for n_{dof}=2, 3, and 6 degrees of freedom per interface and n_p=1, 3, and 6 load cases, respectively. Each load case is studied in the context of one optimization, denoted by (1) in Table 4.3. The results of the proposed approach (B) are compared with the results of a monolithic optimization (A) for all design problems (P1.1-3) to investigate optimality or rather convergence to the monolithic solution of (A) and physical feasibility. For (P1.1), also a second decoupled architecture (C) is investigated that does not provide detail level information by meta models to the system level.

Second, in Chapter 6, the Informed Decomposition (B) is then further analyzed by examining the design problem class (P2) in terms of practical application. Thus, a four-component system for n_{dof}=2, 3, and 6 degrees of freedom per interface and corresponding n_p=1, 6, and 100 load cases considered simultaneously, is investigated. In the first investigation (P2.1), the proof of the general applicability of the offline database for components of design domains Ω varying in dimensions is given. Then, in (P2.2), a computational time comparison between approach (B), two monolithic architectures (A1-2), and analytical target cascading (D) is performed. Finally, in (P2.3), we address a design problem involving a low-cost lightweight robot for a pick-and-place task with n_p=100 representative static load cases.

Table 4.3: Overview of the design problem classes (P1) and (P2) with the complexity drivers n_c as the number of components, n_{dof} as the number of interface degrees of freedom, and n_p as the number of load cases, and the different optimization architectures (A) as monolithic architecture, (B) as the proposed Informed Decomposition, (C) uninformed decomposition, and (D) analytical target cascading

	Design problem		Complexity drivers			Optimization architectures			
			n_c	n_{dof}	n_p	(A)	(B)	(C)	(D)
Chapter 5	(P1)	(P1.1)	2	2	1 (1)	x	x	x	
		(P1.2)	2	3	3 (1)	x	x		
		(P1.3)	2	6	6 (1)	x	x		
Chapter 6	(P2)	(P2.1)	4	2	1	x	x		
		(P2.2)	4	3	6	x	x		x
		(P2.3)	4	6	100	x	x		

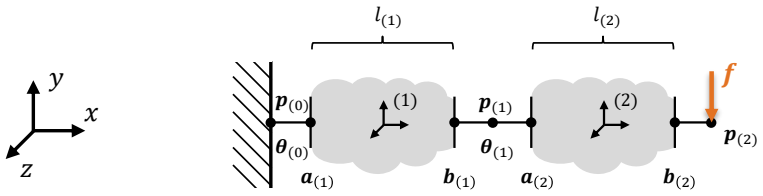
5 Design problem class (P1): Physical feasibility and optimality

5.1 Setup

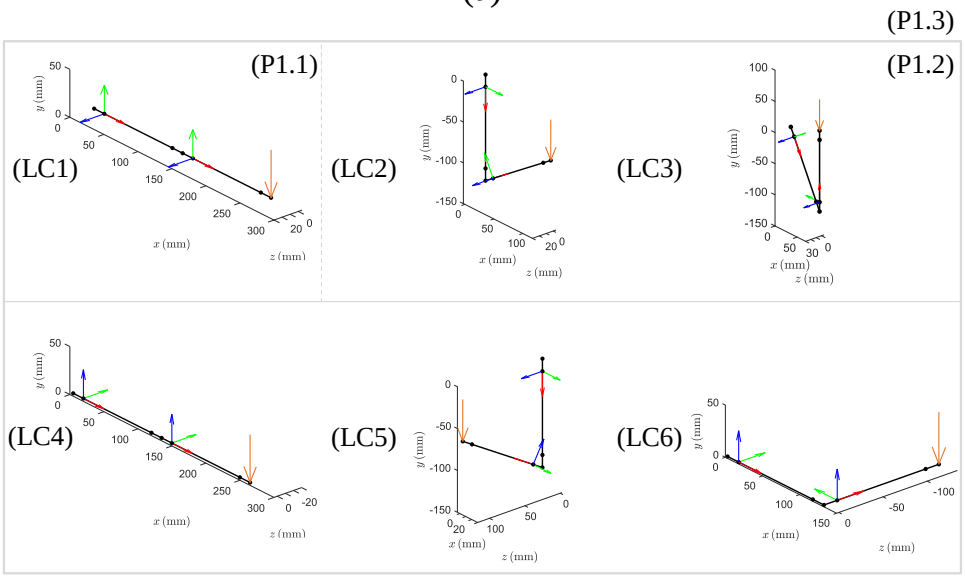
First, a simple two-component system, with a rectangular design domain Ω of width and height $w_{(i)}$=$h_{(i)}$=30 mm, and length $l_{(i)}$=100 mm for both components, is investigated for a required system displacement of up to $u_{\max}$=1 mm. The rectangular geometrical design domain Ω is discretized with $n_{\text{ele}} = [20, 8, 8]^\top$ elements in x, y, and z-direction for each component i. The material used is a synthetic resin for additive manufacturing, see Table 5.1. The given design problem class (P1) is solved for n_{p}=6 different load cases, see Fig. 5.1 and Table 5.2, while each load case is considered within one single optimization and represents a main load in one coordinate direction.

Table 5.1: Material data of a synthetic resin for the components i of both design problem classes (P1) and (P2)

Material	Young's Modulus E (GPa)	Poisson's Ratio ν	Density ρ (g/cm^3)
Synthetic resin	10.0	0.36	1.63



(a)



(b)

Figure 5.1: Design problem class (P1) with (a) a simple two-component system and (b) the n_p=6 respective load cases that are investigated. (P1.1) covers load case (LC1), (P1.2) covers load cases (LC1-3) and (P1.3) covers all load cases (LC1-6)

Table 5.2: Definition of the n_p=6 load cases (LC) of design problem class (P1) with a main shear force in y (y-shear), bending moment about the z-axis (z-bending), normal force in x (x-normal), shear force in z (z-shear), bending moment about the y-axis (y-bending) and torsional moment about the x-axis (x-torsion)

Load case	Main load	$\boldsymbol{\theta}_{(0)}$ rad	$\boldsymbol{\theta}_{(1)}$ rad	$\boldsymbol{f}$ N	$\boldsymbol{p}_{(0)}$ mm	$\boldsymbol{a}_{(1)}$ mm	$\boldsymbol{b}_{(1)}$ mm	$\boldsymbol{p}_{(1)}$ mm	$\boldsymbol{a}_{(2)}$ mm	$\boldsymbol{b}_{(2)}$ mm	$\boldsymbol{p}_{(2)}$ mm
(LC1)	y-shear (2)	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-100\end{bmatrix}$							
(LC2)	z-bending (6)	$\begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\\\frac{4\pi}{6}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-300\end{bmatrix}$							
(LC3)	x-normal (1)	$\begin{bmatrix}0\\0\\-\frac{2\pi}{6}\end{bmatrix}$	$\begin{bmatrix}0\\0\\\frac{5\pi}{6}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-3000\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}15\\0\\0\end{bmatrix}$	$\begin{bmatrix}115\\0\\0\end{bmatrix}$	$\begin{bmatrix}130\\0\\0\end{bmatrix}$	$\begin{bmatrix}145\\0\\0\end{bmatrix}$	$\begin{bmatrix}245\\0\\0\end{bmatrix}$	$\begin{bmatrix}260\\0\\0\end{bmatrix}$
(LC4)	z-shear (3)	$\begin{bmatrix}-\frac{\pi}{2}\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-100\end{bmatrix}$							
(LC5)	y-bending (5)	$\begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\-\frac{4\pi}{6}\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-300\end{bmatrix}$							
(LC6)	x-torsion (4)	$\begin{bmatrix}-\frac{\pi}{2}\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-100\end{bmatrix}$							

The respective monolithic optimization problem (1.3) is now decoupled utilizing the Informed Decomposition approach of Chapter 4. As introduced in Section 4.1.2, symmetry conditions are enforced leading to the reduced κ-representation (4.21) with eight entries $\boldsymbol{\kappa}=\boldsymbol{\kappa}_{\text{sym}} \in \mathbb{R}^{1\times 8}$. Since the length $l_{(i)}$ of both components is the same, the meta models are reduced exclusively to $\boldsymbol{\kappa}$ as the input

$$\hat{p}(\boldsymbol{\kappa}), \tag{5.1}$$

$$\hat{m}(\boldsymbol{\kappa}). \tag{5.2}$$

The classical monolithic optimization (1.3) referred to as (A), serves as a benchmark. The main objective of this chapter is to show validity of the developed Informed Decomposition approach (B) by analyzing the

1. physical feasibility p and
2. optimality m.

This is done by slowly increasing complexity for the first design problem class (P1) and comparing the respective results with (A) by investigating the system with

(P1.1) n_{p}=1 load case (LC1) for a straight pose with two degrees of freedom per interface n_{dof}=2 and an additional comparison to the uninformed decomposition approach (C),
(P1.2) n_{p}=3 planar load cases (LC1-3) with three degrees of freedom per interface n_{dof}=3, and
(P1.3) n_{p}=6 load cases (LC1-6) with all six degrees of freedom per interface n_{dof}=6.

5.2 Design problem (P1.1): Load case in straight pose

5.2.1 Introduction

For the first design problem (P1.1), the load case (LC1) is investigated. The proposed Informed Decomposition (B) is hereby not only compared to the results of monolithic optimization (A), but also to a decoupled uninformed decomposition approach (C). Since only translational displacements in the force direction y and rotations about the z-axis occur for a linear investigation in a straight pose, the interface description can be reduced to n_{dof}=2. Namely, the shear degrees of freedom in the y-axis $(2, 8)$ and the rotational degrees of freedom about the z-axis $(6, 12)$, see Fig. 5.2. Note that the following section is based on partial results of a similar design problem that was investigated in Krischer & Zimmermann (2021) and during a supervised student's project in Kerscher (2021).

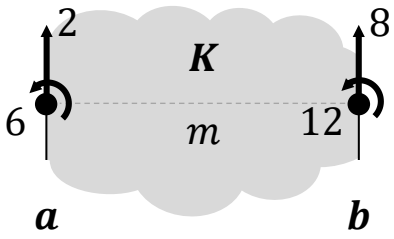


Figure 5.2: A three-dimensional mechanical body with n_{int}=2 interfaces and n_{dof}=2 degrees of freedom per interface

The κ-representation is hence reduced to

$$\boldsymbol{\kappa}_{\text{sym}} = \quad [k_{2,2}, k_{6,6}, k_{12,12}] \quad \in \mathbb{R}^{1\times 3}, \tag{5.3}$$

and the corresponding interface stiffness matrix is

$$\boldsymbol{K} \in \mathbb{R}^{4\times 4}. \tag{5.4}$$

5.2.2 Offline database

For (P1.1) with n_{dof}=2, no meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ exist in the offline database. Therefore, the active-learning strategy from Section 4.3 is used to establish new meta models. The sampling parameters can be obtained from Table 5.3.

Table 5.3: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P1.1)

(i) Classification sampling: $N_J = [100, 200, \ldots, 1100]$ for $J = 1, \ldots, 10$			
Physical seed $\boldsymbol{Y}_{\text{p}}$	Classification samples $\boldsymbol{Y}_{\text{C}}$	Temporary samples $\boldsymbol{Y}_{\text{t}}$	Subset of samples $\boldsymbol{Y}_{\text{s}}$
$N_{\text{p}} = 100$	$N_{\text{C}} = 1100$	$N_{\text{t}} = J^4 \frac{\left(10^6 - 10^3\right)}{10^4} + 10^3$	$N_{\text{s}} = 100$
(ii) Regression sampling: for $A = [650, \ldots, 1500]$			
Feasible classification samples $\boldsymbol{Y}_C = 1$		Regression samples $\boldsymbol{Y}_{\text{R}}$	
$C_{\text{p}} = 649$		$N_{\text{R}} = 1500$	

To illustrate how the active-learning strategy works, Fig. 5.3 shows the iterations J=1 and 2 of the classification phase (i). First, in Fig. 5.3 (a), the initial physically feasible seed is shown $\boldsymbol{Y}_\mathrm{p}$. The physical seed corresponds to actual physical topologies that are created with prescribed densities $\boldsymbol{x}$=$\boldsymbol{\rho}$ and reduced to $\boldsymbol{\kappa}$ using the reduction procedure of Section 4.1.1. Hence, it is known that these $\boldsymbol{\kappa}$ exist. Next, randomly generated temporary samples $\boldsymbol{Y}_\mathrm{t}$ are calculated for iteration J=1, Fig. 5.3 (b), and the sub samples $\boldsymbol{Y}_\mathrm{s}$ closest to the feasible points are selected, Fig. 5.3 (c). Then, all selected sample points are evaluated using component optimization (4.37) and iteration J=2 begins. Fig. 5.3 (d) shows the separation plane of the temporary binary SVM $\hat{p}_2$. It can be seen that the shape is irregular, which is due to the small number of sample points. Temporary samples $\boldsymbol{Y}_\mathrm{t}$ are then created again, Fig. 5.3 (e), and the points $\boldsymbol{Y}_\mathrm{s}$ that are now closest to the new SVM separation plane are selected, Fig. 5.3 (f). This process is performed for all iterations $J = 1, ..., 10$. During the active learning strategy, the shape and hence the quality of the SVM will successively improve.

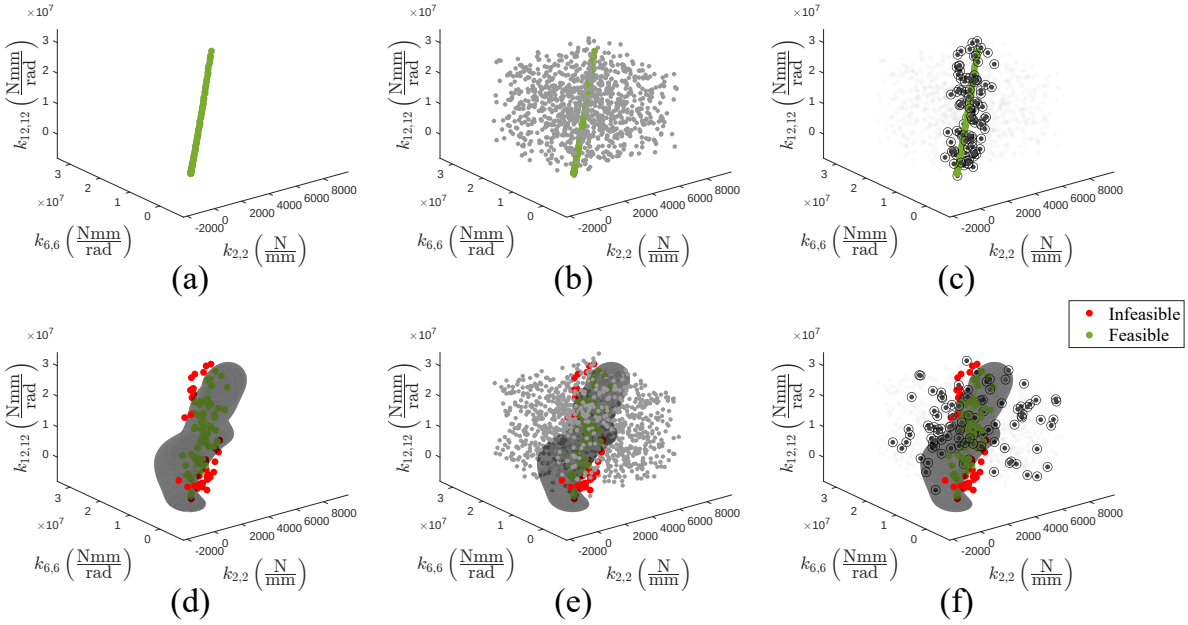


Figure 5.3: Iteration J=1: (a) Initial physical seed $\boldsymbol{Y}_p$, (b) temporary samples $\boldsymbol{Y}_t$, (c) chosen subset of classification samples $\boldsymbol{Y}_s$. Iteration J=2: (d) Visualization of the SVM for the samples $\boldsymbol{Y}_p$ of the first iteration J=1, (e) temporary samples $\boldsymbol{Y}_t$, (f) chosen subset of classification samples $\boldsymbol{Y}_s$. Circles indicate proximity to the feasibility boundary

The sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ computed by the active-learning strategy is shown in Fig. 5.4 (i)-(ii). Fig. 5.4 (i) shows the infeasible data points outside the physically feasible design space, while all feasible data points are inside the SVM plane. Fig. 5.4 (ii) shows the physically feasible data points inside the SVM plane generated in both phases (i) and (ii). The general shape of the physically feasible region is a three-dimensional solid that tapers towards high or low stiffness values. One can observe a smooth transition of the volume fraction from data points with low stiffness values and also low volume fraction values to designs with high stiffnesses and correspondingly high volume fraction values. In addition, the volume fraction of stiffness designs increases as it approaches the boundary of the physically feasible design space. The active-learning strategy almost captured the lower and upper bound of the design space, corresponding to a completely void and filled design domain Ω. It should be noted that some physically feasible data points are outside the SVM plane. This is because different feasibility estimators $\hat{p}_J(\boldsymbol{\kappa})$, for $J = 1, ..., 10$, are trained during the active-learning strategy and therefore the shape of the separation plane may also change slightly, causing earlier feasible points to be falsely classified as infeasible.

The sample data can then be used to train the final meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$. The feasibility estimators $\hat{p}_J(\boldsymbol{\kappa})$ for $J = 2, ..., 10$ are realized with a binary SVM classifier from MATLAB with radial basis functions as the nonlinear kernel function $\phi(\boldsymbol{y})$. For the final estimator $\hat{p}$=$\hat{p}_{10}$, for J=10, the data is split into 80% training and 20% test samples. The hyperparameters are a kernel scale parameter k_s and the box-constraint C, which are determined using a Bayesian optimization with a five-fold cross-validation on the training data. To assess the performance of the classifier, the earlier introduced FPR, TPR, and ACC were analyzed, see Table 5.4. Since a decision for an infeasible design on the system level makes the whole design approach invalid, the most important quantity is a low FPR. The final separating plane $\hat{p}$ can once again be seen in Fig. 5.4. Compared to the temporary classifier $\hat{p}_2(\boldsymbol{\kappa})$ from Fig. 5.3 (d), the volume of the physically feasible space is larger, and the shape of the plane is more regular.

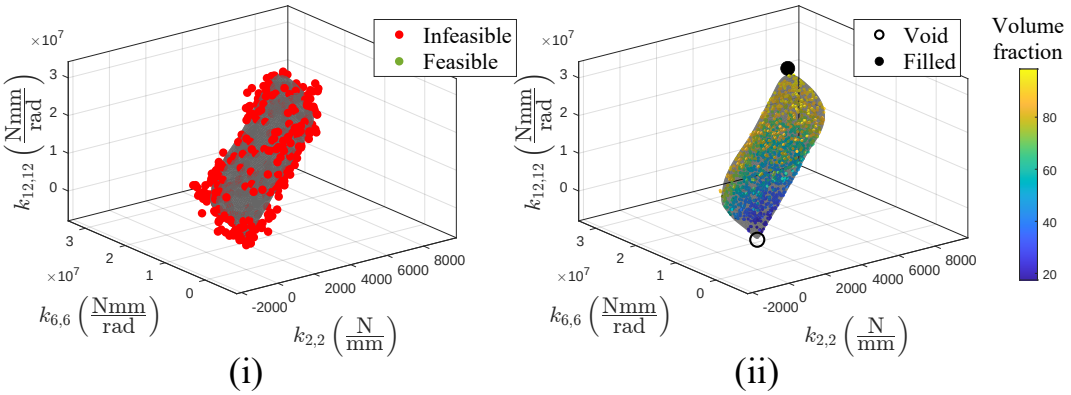


Figure 5.4: Results of the active-learning strategy for (P1.1): (i) the final SVM separation plane of $\hat{p}_{10}$ and the respective infeasible classification samples $[\boldsymbol{Y}, \boldsymbol{Z}]_C$ and (ii) the regression samples $[\boldsymbol{Y}, \boldsymbol{Z}]_M$ within the physically feasible design domain

Then, the mass estimator $\hat{m}(\boldsymbol{\kappa})$ is implemented with a feedforward artificial neural network (ANN). The sample data is divided into 80% training, 10% validation and 10% test samples. The mass estimator $\hat{m}(\boldsymbol{\kappa})$ is trained using the Levenberg-Marquardt algorithm in MATLAB. To select the best configuration of hidden layers and corresponding neurons, a grid search for $n_{\text{hl}} = 1, ..., 5$ hidden layers and $n_{\text{n}} = 1, ..., 10$ neurons was performed. The configuration with the lowest R^2-value was chosen. The final ANN has n_{hl}=3 hidden layers and n_{n}=7 hidden neurons. In addition, the MSE of the volume fraction was taken as a second performance measure, see Table 5.4.

Table 5.4: Performance measures of $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for (P1.1)

$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$	
FPR	TPR	ACC	R^2	MSE (%)
0.0286	0.973	0.973	0.996	1.61

Having created the meta models needed for the offline database, approach (B) is now ready to optimize (P1.1) by decomposing the original monolithic optimization problem (1.3) of (A).

5.2.3 Uninformed decomposition (C)

In addition to (A) and (B), also a third IO formulation (C) is investigated in this section. The so-called uninformed decomposition approach (C) has been developed in Krischer et al. (2020) for a similar two-component structure. The name uninformed decomposition refers to decomposition without providing detail level information (III) by meta models to the system level (I).

The system is decoupled based on assumptions, e.g., by a reference structure that is not necessarily optimal. Then, each component is optimized with respect to its own local objectives, design variables and constraints. Thus, this is also a decoupled approach, with the important difference from (B) that no estimate $\hat{m}$ of the mass is used to balance the requirements on the stiffness matrices of the individual components, nor is any estimate of the physical feasibility $\hat{p}$ used, potentially affecting both optimality and physical feasibility.

The total translational deformation

$$u_{\text{ee}} = u^p_{(1)} + \varphi^p_{(1)} l_{(2)} + u^p_{(2)}, \tag{5.5}$$

can be additively decomposed into the translational displacement $u^p_{(1)}$ and the rotational displacement $\varphi^p_{(1)}$ of the first component and translational displacement $u^p_{(2)}$ of the second component, see Fig. 5.5.

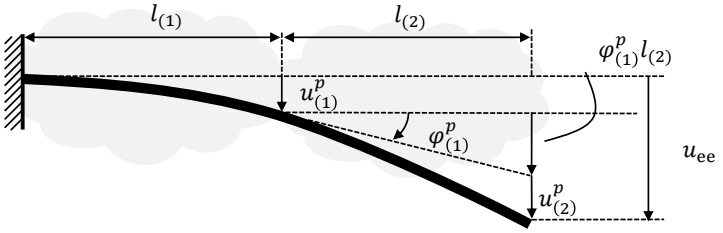


Figure 5.5: Additive decomposition of the end effector displacement u_{ee} into the respective component displacements $u^p_{(1)}$, $\varphi^p_{(1)}$, and $u^p_{(2)}$

The system requirement $u_{\text{ee}} \leq u_{\max}$ will be satisfied, whenever the following component requirements are satisfied

$$b_{(1)}(\boldsymbol{x}) = \left(u^p_{(1)} + \varphi^p_{(1)} l_{(2)}\right) - \alpha u_{\max} \leq 0, \tag{5.6}$$

$$b_{(2)}(\boldsymbol{x}) = u^p_{(2)} - (1-\alpha)\, u_{\max} \leq 0, \tag{5.7}$$

where α can assume any value between 0 and 1. α controls how much stiffness is required of each component in order to meet the system stiffness requirement. Two stiffness distributions are chosen for comparison: α=0.5, (C1), and α=0.6, (C2).

The component optimization for the uninformed decomposition (C) of each component reads

$$\begin{aligned} &\min_{\boldsymbol{x}_{(i)}} m_{(i)}(\boldsymbol{x}_{(i)}), \\ \text{s. t.:} \quad & b_{(i)}(\boldsymbol{x}_{(i)}) \leq 0, \\ & \boldsymbol{x}_{\text{lb}} \leq \boldsymbol{x}_{(i)} \leq \boldsymbol{x}_{\text{ub}}. \end{aligned} \tag{5.8}$$

Having established all approaches (A), (B), and (C1-2), the design problem (P1.1) can now be solved.

5.2.4 System optimization

First, the quantities of interest $z = [m, u]$ on the system level (I) are computed in terms of the level (II) κ-representations. Thus, the monolithic approach (A) is executed, and the corresponding $\boldsymbol{\kappa}$-values are computed after the optimization has converged. For the decoupled architecture of the Informed

Decomposition (B), the surrogate-based system optimization (4.30) is solved with $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ using PSO. Finally, the uninformed decomposition (C) of (5.8) is used, for α=0.5 and α=0.6, and the respective $\boldsymbol{\kappa}$ are computed after convergence of the component optimizations. All $\boldsymbol{\kappa}_{(i)}$ vectors for the utilized approaches (A), (B), and (C1-2) can be seen in Table 5.5. The monolithic approach (A) and the Informed Decomposition (B) yield designs for each component i=1, 2 with similar stiffness values. In contrast, the uninformed approaches (C1) and (C2) show higher stiffness values for the first component and lower stiffness values for the second component, indicating a suboptimal stiffness distribution.

Table 5.5: Component performance (II), $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$, and the quantities on the system level (I), $\boldsymbol{z} = [m, u]$, of design problem (P1.1) for monolithic optimization (A), the proposed Informed Decomposition (B), and the uninformed decomposition with α=0.5, (C1), and with α=0.6, (C2)

Component-performance level (II): $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$						
				$\boldsymbol{\kappa}_{(i)}$		
Load case	Comp. (i)	Approach	$v_{(i)}$	$k_{2,2}$	$k_{6,6}$	$k_{12,12}$
			%	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$
(LC1)	(1)	(A)	51.48	1.19e3	9.04e6	8.04e6
		(B)	48.39	1.37e3	9.46e6	7.57e6
		(C1)	100.0	7.17e3	2.51e7	2.51e7
		(C2)	69.35	2.61e3	1.37e7	1.26e7
	(2)	(A)	27.84	588	5.15e6	1.42e6
		(B)	31.24	973	6.46e6	3.10e6
		(C1)	22.36	306	2.66e6	8.49e5
		(C2)	23.81	398	3.38e6	9.25e5

Quantities on the system level (I): $\boldsymbol{z} = [m, u]$					
Load case		Approach	$\sum m_{(i)}$	$\frac{(\cdot)-m_A}{m_A}$	u
			g	%	mm
(LC1)		(A)	114.0	-	1.00
		(B)	114.5	0.40	0.99
		(C1)	175.9	54.3	1.04
		(C2)	133.9	17.5	1.00

5.2.5 Component optimization

The system optimization (4.30) of approach (B) has decoupled the given design problem enabling n_c=2 independent component optimizations (4.37). Each optimized $\boldsymbol{\kappa}_{(i)}$ of level (II) can be taken as a target stiffness for component optimization (4.37) to compute the final topologies $\boldsymbol{x}_{(i)}$ at level (III). The results are then compared to (A) and (C1-2) and are listed in Table 5.5 together with the resulting quantities of interests on the system level (I), $\boldsymbol{z} = [m, u]$. Since the system optimization and the component optimization operate on volume fractions $v_{(i)}$ and $\boldsymbol{\kappa}_{(i)}$, the component-performance level quantities (II) are given as $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$.

All approaches (A), (B), and (C1-2) produced qualitatively similar topologies for load case (LC1) in straight pose for both components, see Fig. 5.6. The first component experiences a mixed load case consisting of a z-bending moment and a y-shear force. Therefore, on the one hand, the material is distributed farthest from the main bending axis, while on the other hand, a slightly tapered tip in the beginning with reinforcements in the center for the occurring shear loads can be seen. In contrast, the second component shows a classical cantilever beam topology with a more pronounced tip, where

the y-shear force is applied, and reinforcements in the center between the upper and lower side of the component. The difference between component i=1 and i=2 is only the ratio between shear force and bending moment. The proposed approach (B) produces a slightly lighter first component compared to the reference design (A), while the second component is slightly heavier. Overall, the total mass achieved by the Informed Decomposition is 0.40% heavier than the reference design resulting from (A), while both meet the requirement $u \leq u_{\max}$ for system stiffness. The approaches (C1) and (C2) show in accordance with the higher stiffness values, also higher masses for the first component, and lower masses for the lower stiffnesses of the second component. Overall, both approaches (C1) and (C2) yield system masses that are far from the benchmark results of (A), with a deviation of 54.3% and 17.5%, respectively. In addition, the first component of (C1) is completely filled with material, which is due to an overly demanding requirement, which therefore cannot be met because it is not physically feasible (p=−1). Hence, also the system requirement $u \leq u_{\max}$ is violated with u=1.04 mm for (C1), while the approaches (A), (B), and (C2) are all feasible.

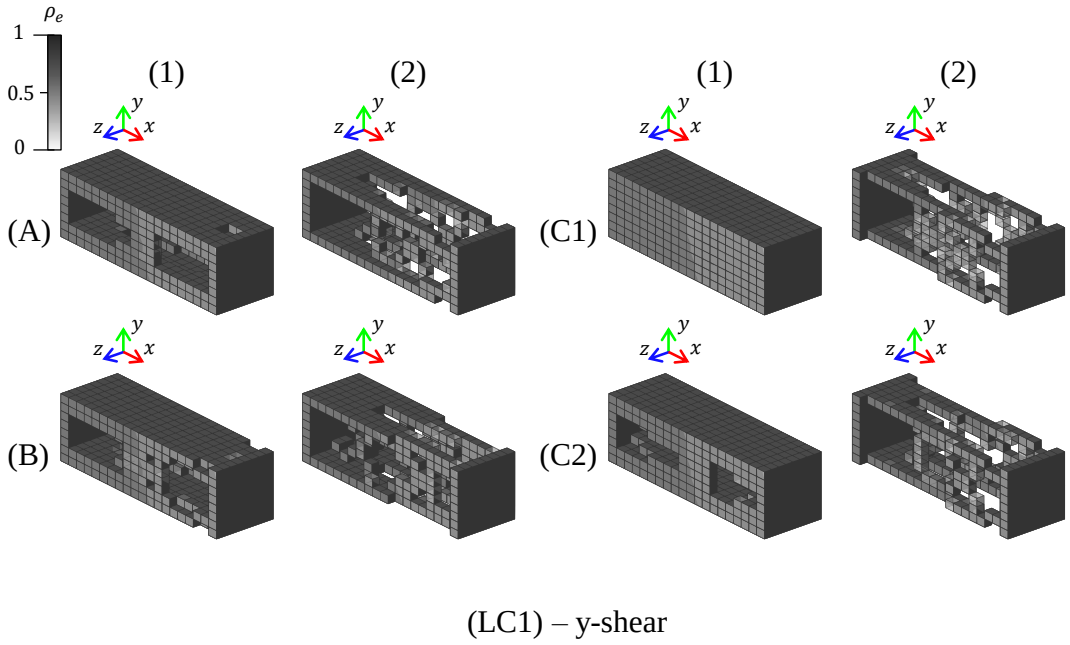


Figure 5.6: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P1.1) of monolithic optimization (A), Informed Decomposition (B), uninformed decomposition with α=0.5, (C1), and α=0.6, (C2)

5.3 Design problem (P1.2): Planar load cases

5.3.1 Introduction

For the second design problem (P1.2), the load cases (LC1-3) are analyzed. Unlike the previous section, now only a comparison between the monolithic optimization (A) and the Informed Decomposition (B) is carried out. The planar load cases demanding for all planar degrees of freedom of the components i. Therefore, the stiffness matrix is reduced to n_{dof}=3 with the normal degrees of freedom $(1, 7)$, shear degrees of freedom in the y-axis direction $(2, 8)$, and the rotational degrees of freedom about the z-axis $(6, 12)$, see Fig. 5.7. It should be noted that research addressing a similar design problem was conducted in a supervised student's project of Li (2022) and this section is based on partial results from that.

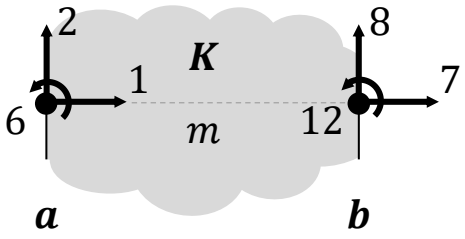


Figure 5.7: A three-dimensional mechanical body with n_{int}=2 interfaces and n_{dof}=3 degrees of freedom per interface

The κ-representation is hence reduced to

$$\boldsymbol{\kappa}_{\text{sym}} = \quad [k_{1,1}, k_{2,2}, , k_{6,6}, k_{12,12}] \quad \in \mathbb{R}^{1\times 4}, \tag{5.9}$$

and the corresponding interface stiffness matrix is

$$\boldsymbol{K} \in \mathbb{R}^{6\times 6}. \tag{5.10}$$

5.3.2 Offline database

The monolithic optimization (A) of (P1.2) does not require any further preparation. In contrast, the offline database of approach (B) does not contain any meta model $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for n_{dof}=3. Therefore, the offline database is extended by utilizing the active-learning sampling strategy of Section 4.3. The parameters for the active-learning sampling are documented in Table 5.6.

Table 5.6: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P1.2)

(i) Classification sampling: $N_J = [600, 1200, ..., 6600]$ for $J = 1, ..., 10$			
Physical seed $\boldsymbol{Y}_{\text{p}}$	Classification samples $\boldsymbol{Y}_{\text{C}}$	Temporary samples $\boldsymbol{Y}_{\text{t}}$	Subset of samples $\boldsymbol{Y}_{\text{s}}$
$N_{\text{p}} = 600$	$N_{\text{C}} = 6600$	$N_{\text{t}} = J^4 \frac{\left(10^6 - 10^5\right)}{10^4} + 10^5$	$N_{\text{s}} = 600$
(ii) Regression sampling: for $A = [4301, ..., 8000]$			
Feasible classification samples $\boldsymbol{Y}_C = 1$		Regression samples $\boldsymbol{Y}_{\text{R}}$	
$C_{\text{p}} = 4300$		$N_{\text{R}} = 8000$	

The sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ computed by the active-learning strategy is visualized in Fig. 5.8, while Fig. 5.8 (i) shows the classification sample data and Fig. 5.8 (ii) the created regression samples. Since the stiffness design space is now $\mathbb{R}^4$, the data is projected onto two two-dimensional scatter plots to show the physically feasible design space. Because this section and the previous one use the same rectangular design domain Ω of width and height $w_{(i)}$=$h_{(i)}$=30 mm, and length $l_{(i)}$ = 100 mm, the shared stiffness values $k_{2,2}$,$k_{6,6}$, and $k_{12,12}$ have in theory the same lower and upper bounds. Yet, the statistic nature of the active-learning strategy that is also influenced by the number of dimensions n_k cannot ensure that the complete design space is sampled. Nevertheless, the comparison with the upper (filled) and lower limit (void) of the diagonal stiffness values, indicates that the active-learning strategy again almost captured both bounds. The resulting sample data can then be used to train the final meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ in accordance with Section 5.2. The performance measures for both estimators are listed in Table 5.7.

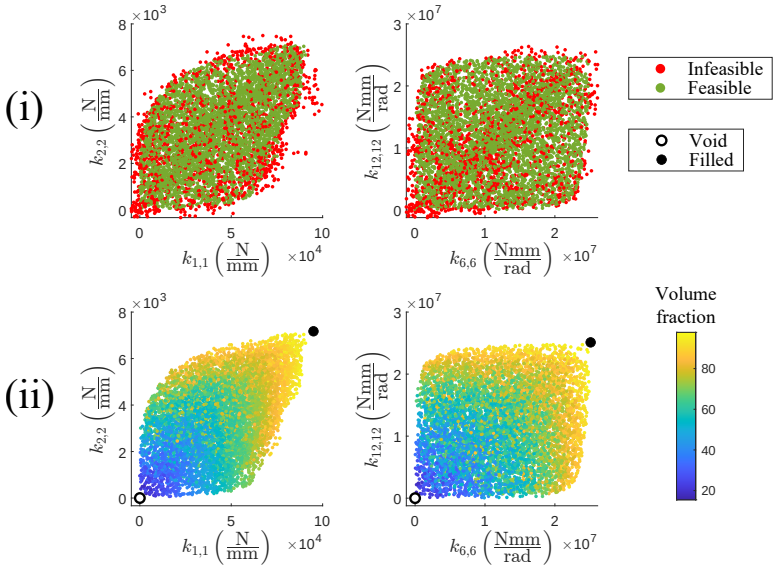


Figure 5.8: Results of the active-learning strategy for (P1.2): (i) classification samples for physical feasibility estimation and (ii) regression samples for mass estimation

Table 5.7: Performance measures of $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for (P1.2)

$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$	
FPR	TPR	ACC	R^2	MSE (%)
0.0355	0.920	0.931	0.992	2.31

5.3.3 System optimization

With the available meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ in the offline database, the investigation of (P1.2) can be started. Approach (A) and (B) are applied to the three load cases (LC1-3) of the design problem (P1.2). First, the level (II) κ-representations are computed and the resulting $\boldsymbol{\kappa}_{(i)}$ can be seen in Table 5.8. The monolithic approach (A) and the Informed Decomposition (B) produce similar $\boldsymbol{\kappa}_{(i)}$ designs for both components and all load cases (LC1-3), while no significant outlier can be identified.

Table 5.8: Component performance (II), $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$, and the quantities on the system level (I), $\boldsymbol{z} = [m, u]$, of design problem (P1.2) for monolithic optimization (A) and the proposed Informed Decomposition (B)

Component-performance level (II): $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$							
				$\boldsymbol{\kappa}_{(i)}$			
Load case	Comp. (i)	Approach	$v_{(i)}$	$k_{1,1}$	$k_{2,2}$	$k_{6,6}$	$k_{12,12}$
			%	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$
(LC1)	(1)	(A)	51.48	4.12e4	1.19e3	9.04e6	8.04e6
		(B)	47.68	3.51e4	1.91e3	1.05e7	9.19e6
	(2)	(A)	27.84	1.11e4	588	5.15e6	1.42e6
		(B)	31.18	1.32e4	859	6.69e6	1.88e6
(LC2)	(1)	(A)	55.04	4.77e4	399	7.26e6	7.26e6
		(B)	53.60	4.03e4	615	7.07e6	7.44e6
	(2)	(A)	37.02	2.05e4	1.09e3	8.44e6	2.95e6
		(B)	42.06	2.66e4	1.89e3	1.08e7	5.81e6
(LC3)	(1)	(A)	68.41	4.81e4	1.09e3	1.27e7	2.21e6
		(B)	66.92	4.26e4	1.18e3	1.26e7	3.42e6
	(2)	(A)	30.62	1.97e4	1.02e3	3.14e6	3.14e6
		(B)	33.95	2.40e4	880	2.25e6	3.74e6

Quantities on the system level (I): $\boldsymbol{z} = [m, u]$					
Load case		Approach	$\sum m_{(i)}$	$\frac{(\cdot) - m_A}{m_A}$	u
			g	%	mm
(LC1)		(A)	114.0	-	1.00
		(B)	113.4	−0.57	0.99
(LC2)		(A)	132.4	-	1.00
		(B)	137.5	3.91	1.00
(LC3)		(A)	142.4	-	1.00
		(B)	145.0	1.87	1.00

5.3.4 Component optimization

Then, the optimized $\boldsymbol{\kappa}_{(i)}$ of level (II) are taken as target values to compute the final topologies $\boldsymbol{x}_{(i)}$ at level (III) using the decoupled component optimizations (4.37) of the proposed approach (B). The resulting topologies are shown in Fig. 5.9, while the respective volume fractions $v_{(i)}$ and the resulting quantities $\boldsymbol{z} = [m, u]$ on the system level (I) are listed in Table 5.8.

For the already studied y-shear load case (LC1), in theory the same two-component cantilever topology should be produced as in Section 5.2. This is indeed the case for the monolithic approach (A), however, the proposed approach (B) has a slightly lighter first component, while the second component is slightly heavier. However, the total mass of (B) is 0.57% less than that of the reference design (A). Since both approaches meet the system requirements for displacement, both approaches are feasible. The fact that (B) slightly outperforms the reference solution (A) can be explained by the non-convexity property of the topology optimization design problem, which does not guarantee global convergence for arbitrary start vectors.

The z-bending load case (LC2), see Fig. 5.1 (b), yields a topology for the first component that is mainly influenced by a pure bending moment about the local z-axis. Therefore, for (A), the material is distributed only at the boundary of the design domain Ω to compensate for the bending normal stresses there, hence resembling a classical i-beam without web. For the second component of (LC2), a mixture of shear and bending loads is applied, resulting in a similar topology to (LC1). Approach (B)

shows similar results for the first component, but the second component is slightly heavier, resulting in a 3.91% increase in system mass while meeting the system requirement.

The last load case (LC3) is the x-normal load case. A pure normal force is applied to the vertical second component, resulting in a bar-shaped topology in approach (A), while the first component experiences both shear and normal loads. In comparison, approach (B) has problems generating a similar topology for the second component. It yields an asymmetric beam with a higher component mass indicating non-optimality, while the first component looks similar to (A). However, the system mass is only 1.87% higher than (A) with a feasible system design.

In summary, all load cases deviate below 4% from the reference system masses of (A) and are all physically feasible and also feasible in terms of system stiffness $u \leq u_{\max}$. It should be noted that for the three investigated load cases (LC1-3), always the second component with the lower volume fraction shows higher deviations. This could hint to a degradation in the performance of meta models for stiffnesses with low volume fraction.

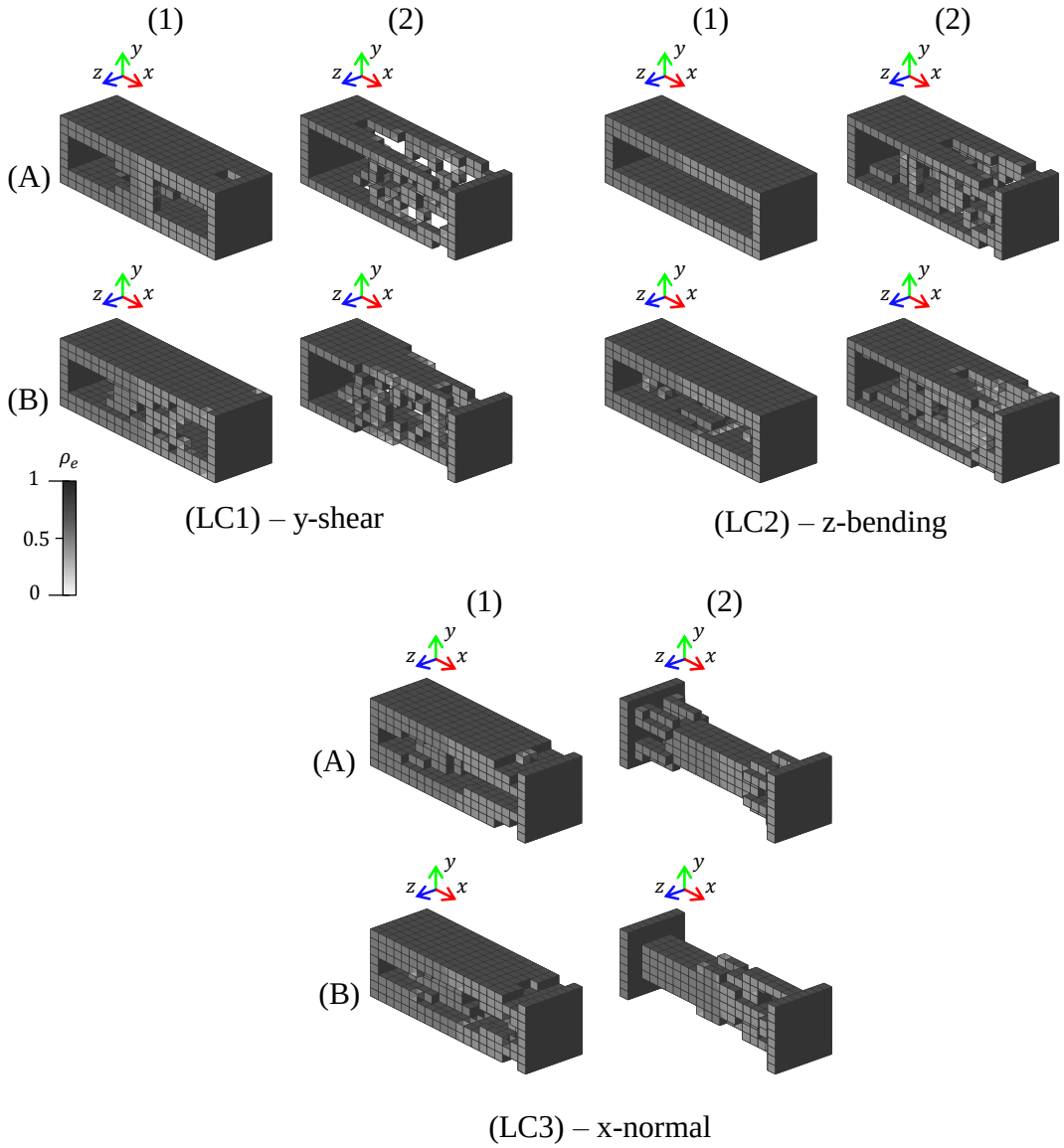


Figure 5.9: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P1.2) of the monolithic optimization (A) and Informed Decomposition (B)

5.4 Design problem (P1.3): Three-dimensional load cases

5.4.1 Introduction

For the last problem (P1.3), all load cases (LC1-6) are analyzed and the results of (B) are compared to the monolithic approach (A). Hence, the complete deformation characteristics need to be considered with the stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ for n_{dof}=6, and the known full κ-representation

$$\boldsymbol{\kappa}_{\text{sym}} = \quad [k_{1,1}, k_{2,2}, k_{3,3}, k_{4,4}, k_{5,5}, k_{6,6}, k_{11,11}, k_{12,12}] \quad \in \mathbb{R}^{1\times8}.$$

5.4.2 Offline database

First, two new meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for the offline database of the Informed Decomposition (B) have to be established. The parameters for the active-learning sampling are shown in Table 5.9.

Table 5.9: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P1.3)

(i) Classification sampling: $N_J = [2000, 2500, 3000, \ldots, 9000]$ for $J = 1, \ldots, 14$			
Physical seed $\boldsymbol{Y}_{\text{p}}$	Classification samples $\boldsymbol{Y}_{\text{C}}$	Temporary samples $\boldsymbol{Y}_{\text{t}}$	Subset of samples $\boldsymbol{Y}_{\text{s}}$
$N_{\text{p}} = 2000$	$N_{\text{C}} = 9000$	$N_{\text{t}} = J^4 \frac{\left(10^7 - 10^4\right)}{14^4} + 10^4$	$N_{\text{s}} = 500$
(ii) Regression sampling: for $A = [2694, \ldots, 8000]$			
Feasible classification samples $\boldsymbol{Y}_C = 1$		Regression samples $\boldsymbol{Y}_{\text{R}}$	
$C_{\text{p}} = 2693$		$N_{\text{R}} = 8000$	

The sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ computed by the active-learning strategy is shown in Fig. 5.10. In comparison to Fig. 5.8, the projection of the scatter plots shows a smaller covered feasible design space for the shared stiffness values $[k_{1,1}, k_{2,2}, k_{6,6}, k_{12,12}]$ of the previous section. This is confirmed by the void and filled design domains Ω which especially for the upper limit is not covered by the sampling strategy.

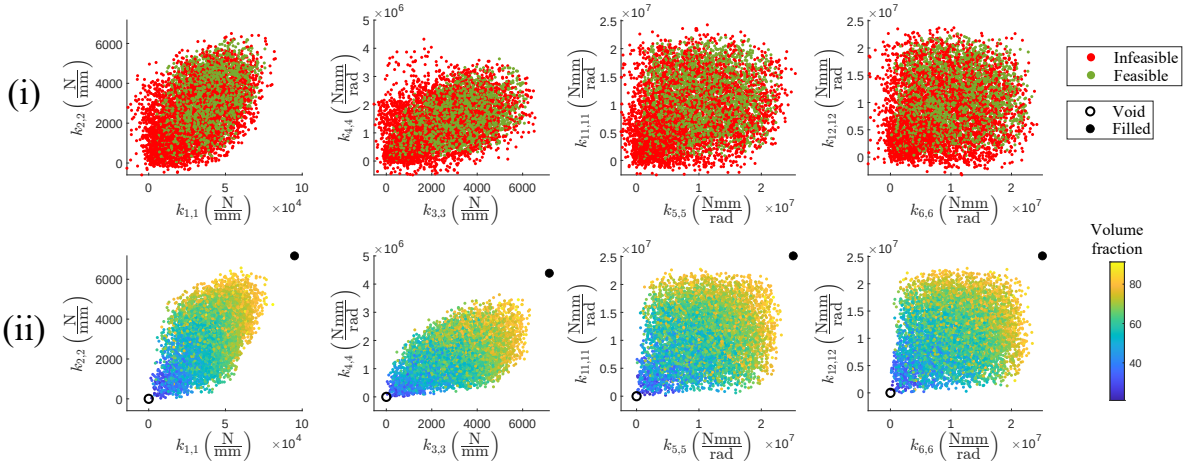


Figure 5.10: Results of the active-learning strategy for (P1.3): (i) classification samples for physical feasibility estimation and (ii) regression samples for mass estimation

The active-learning strategy suffers from the curse of dimensionality, which despite of the higher number of sample points N_C compared to the previous sections prevents the sampling strategy from covering the whole design space. The sample data can then be used to train the final meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$. The performance measurements for both estimators are listed in Table 5.10.

Table 5.10: Performance measures of $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for (P1.3)

$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$	
FPR	TPR	ACC	R^2	MSE (%)
0.0499	0.905	0.927	0.951	7.76

5.4.3 System optimization

The κ-representations of level (II) for the given design problem (P1.3) of the monolithic approach (A) and the system optimization (4.30) of (B) can be seen in Table 5.11. Both (A) and (B) produce results with similar $\boldsymbol{\kappa}_{(i)}$ vectors, yet the deviation of the stiffness values is higher compared to the previous Sections 5.2 and 5.3. In particular one can observe that the system optimization of (B) is not able to realize stiffness values close to the lower bound, e.g., the $k_{2,2}$ and $k_{3,3}$ values for the second component of load case (LC6). Most likely hindered from bad generalization of the meta models that do not have sufficient sample data in this region, hence labeling it as falsely infeasible.

5.4.4 Component optimization

Next, the optimized $\boldsymbol{\kappa}_{(i)}$ of level (II) are utilized for the component optimization of (B) to derive the final topologies $\boldsymbol{x}_{(i)}$ of level (III). The resulting topologies are shown in Fig. 5.11 with the respective volume fractions $v_{(i)}$ and the resulting quantities on the system level (I), $\boldsymbol{z} = [m, u]$, in Table 5.11.

For the load case y-shear (LC1), the Informed Decomposition (B) leads to similar results as (A) with a weight difference of 3.87% from (B), being slightly heavier than in the previous two sections.

The load case (LC2) represents the z-bending case known from the previous section. Here, the first component of approach (B) differs significantly from the one of approach (A) and does not show a clear pure bending characteristic with also a slightly higher volume fraction. The second component of approach (B) is even bulkier than the counterpart of (A), resulting in a total system mass deviation of 8.06%, which is significantly higher than the previous section's result of 3.91%.

The x-normal load case (LC3) also shows significant differences between approach (A) and (B). While component one is similar in both approaches, component two is completely different in appearance. Approach (A) forms the familiar rod-like structure, while approach (B) develops a tube-like design. For the pure normal load case, this is generally not a problem, since what matters is the cross-sectional area provided by each slice of the topology. However, the second component is also considerably heavier and results in the highest deviation of all load cases at 8.18%, which is also in contrast to the previous section's accurate solution of 1.87%.

Load case (LC4) is a z-shear load case, thus the same as (LC1) with an x-rotation of 90°. It is utilized to evaluate approach (B) for the technically same load case but for different stiffness inputs. Since the active-learning strategy automatically samples the design space, small differences are expected to occur for the different dimensions of $\mathbb{R}^8$. While the topologies for approach (A) are the same as for (LC1), only 90° rotated, approach (B) deviates slightly from (LC1), resulting in a slightly higher component mass for the first component and a slightly lower component mass for the second component. The mass deviation compared to (A) is 5.83%, which is worse than the 3.87% of (LC1).

Table 5.11: Component performance (II), $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$, and the quantities on the system level (I), $\boldsymbol{z} = [m, u]$, of design problem (P1.3) for monolithic optimization (A) and the proposed Informed Decomposition (B)

Component-performance level (II): $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$											
				$\boldsymbol{\kappa}_{(i)}$							
Load case	Comp. (i)	Approach	$v_{(i)}$	$k_{1,1}$	$k_{2,2}$	$k_{3,3}$	$k_{4,4}$	$k_{5,5}$	$k_{6,6}$	$k_{11,11}$	$k_{12,12}$
			%	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$
(LC1)	(1)	(A)	51.48	4.12e4	1.19e3	3.26e3	1.19e6	1.20e7	9.04e6	1.17e7	8.04e6
		(B)	47.99	3.37e4	1.78e3	2.50e3	1.19e6	1.06e7	1.09e7	7.51e6	7.63e6
	(2)	(A)	27.84	1.11e4	588	208	1.36e5	1.56e6	5.15e6	2.35e6	1.42e6
		(B)	34.40	1.84e4	1.28e3	1.36e3	7.87e5	6.16e6	7.78e6	3.82e6	3.27e6
(LC2)	(1)	(A)	55.04	4.77e4	399	3.67e3	8.27e5	1.28e7	7.26e6	1.28e7	7.26e6
		(B)	56.29	4.51e4	2.67e3	3.63e3	1.60e6	1.33e7	1.22e7	1.24e7	1.24e7
	(2)	(A)	37.02	2.05e4	1.09e3	772	3.93e5	3.48e6	8.44e6	4.21e6	2.95e6
		(B)	43.20	2.97e4	2.31e3	2.12e3	1.80e6	9.61e6	1.18e7	7.23e6	7.13e6
(LC3)	(1)	(A)	68.41	4.81e4	1.09e3	3.90e3	1.11e6	1.57e7	1.27e7	1.15e7	2.21e6
		(B)	64.33	4.45e4	1.30e3	3.69e3	1.19e6	1.58e7	1.26e7	1.00e7	4.43e6
	(2)	(A)	30.62	1.97e4	1.02e3	1.02e3	2.54e5	3.14e6	3.14e6	3.14e6	3.14e6
		(B)	42.80	3.07e4	1.88e3	1.89e3	1.21e6	7.34e6	7.97e6	7.24e6	7.00e6
(LC4)	(1)	(A)	51.48	4.12e4	3.26e3	1.19e3	1.19e6	9.04e6	1.20e7	8.04e6	1.17e7
		(B)	50.63	3.96e4	3.02e3	2.66e3	2.11e6	1.29e7	1.18e7	1.04e7	1.07e7
	(2)	(A)	27.84	1.11e4	208	588	1.36e5	5.15e6	1.56e6	1.42e6	2.35e6
		(B)	33.30	1.78e4	1.26e3	1.25e3	7.00e5	7.00e6	5.89e6	3.55e6	4.21e6
(LC5)	(1)	(A)	55.04	4.77e4	3.67e3	399	8.27e5	7.26e6	1.28e7	7.26e6	1.28e7
		(B)	54.67	4.38e4	3.78e3	2.20e3	1.66e6	1.12e7	1.34e7	1.09e7	1.27e7
	(2)	(A)	37.02	2.05e4	772	1.09e3	3.93e5	8.44e6	3.48e6	2.95e6	4.21e6
		(B)	43.74	3.12e4	2.44e3	2.24e3	1.71e6	1.18e7	1.08e7	7.10e6	7.75e6
(LC6)	(1)	(A)	49.29	4.09e4	3.51e3	3.51e3	2.89e6	1.39e7	1.39e7	1.28e7	1.28e7
		(B)	46.98	3.77e4	2.89e3	2.84e3	2.59e6	1.21e7	1.21e7	1.03e7	1.04e7
	(2)	(A)	26.43	1.05e4	195	548	1.83e5	4.48e6	1.20e6	1.54e6	2.25e6
		(B)	34.19	1.99e4	1.42e3	1.55e3	9.00e5	7.48e6	6.81e6	4.89e6	4.37e6

Quantities on the system level (I): $\boldsymbol{z} = [m, u]$					
Load case		Approach	$\sum m_{(i)}$	$\frac{(\cdot) - m_A}{m_A}$	u
			g	%	mm
(LC1)		(A)	114.0	-	1.00
		(B)	118.4	3.87	1.00
(LC2)		(A)	132.4	-	1.00
		(B)	143.0	8.06	0.99
(LC3)		(A)	142.4	-	1.00
		(B)	154.0	8.18	1.00
(LC4)		(A)	114.0	-	1.00
		(B)	120.7	5.83	1.00
(LC5)		(A)	132.4	-	1.00
		(B)	141.5	6.90	0.99
(LC6)		(A)	108.9	-	1.00
		(B)	116.7	7.20	0.99

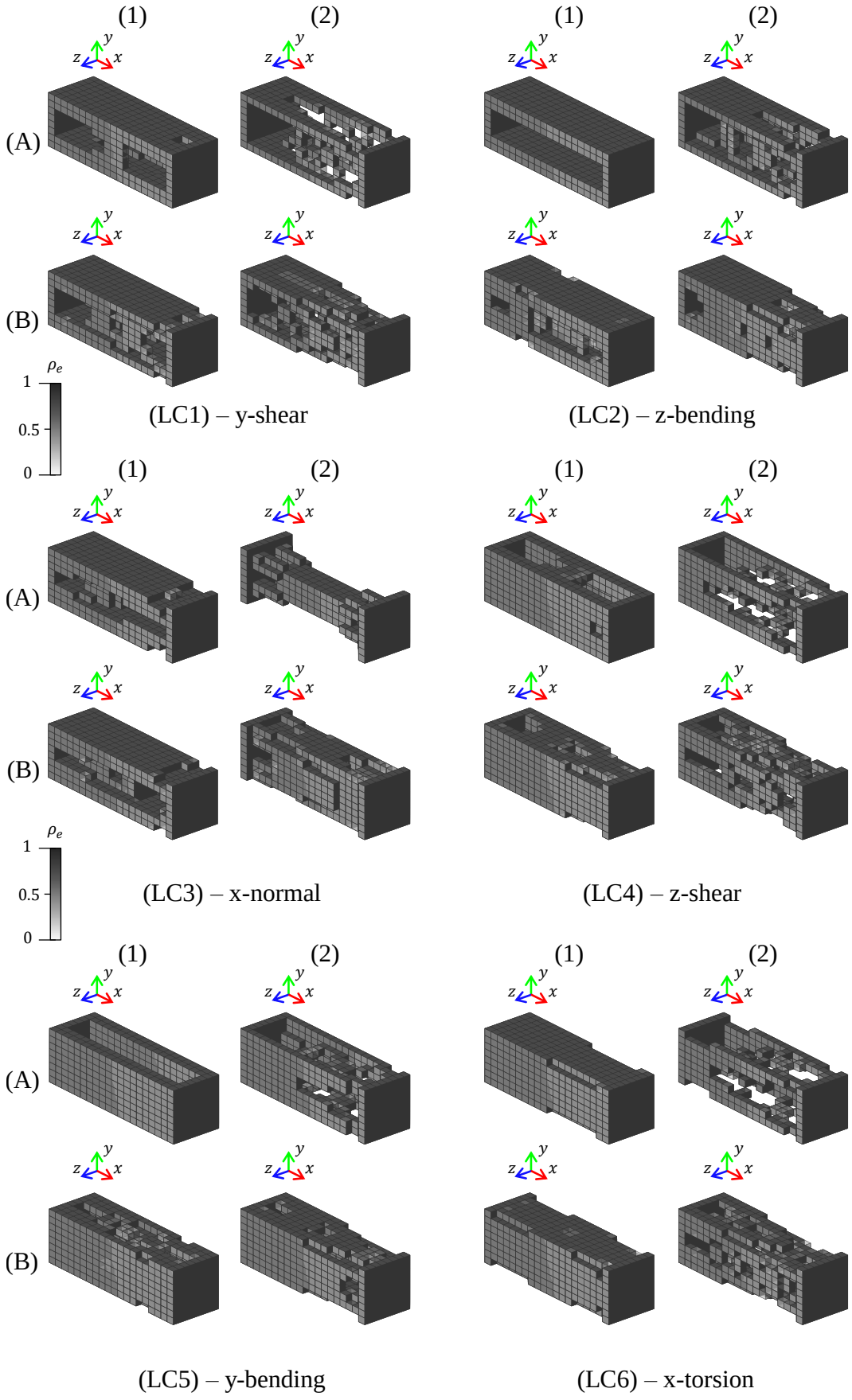


Figure 5.11: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P1.3) of the monolithic optimization (A) and Informed Decomposition (B)

The next load case (LC5) is also a rotated version, but of (LC2), resulting in a bending moment about the y-axis instead of the z-axis. In comparison to (LC2), approach (B) yields better results for the z-bending load case, resulting in a system mass deviation of 6.90% compared to 8.06% of (LC2).

The last load case (LC6) is a torsional load case where the first component is mainly under the influence of torsion while the second component experiences shear forces, also in the z-direction as in (LC4). Approach (A) therefore forms a closed tubular topology for the first component, while the second component has the classical cantilever beam shape. Approach (B) reproduces the topology for the first component quite accurately, but the second component turns out to be somewhat bulkier, with a total mass deviation of 7.20%.

In summary, the Informed Decomposition (B) was able to produce physically feasible designs for all load cases (LC1-6) while satisfying the system stiffness requirement $u \leq u_{\max}$. The mass deviation of the proposed approach (B) has a minimum deviation of 3.87% for the y-shear load case (LC1) and a maximum value of 8.18% for the x-normal load case (LC3). As in the previous section, the second components with the lower volume fractions showed higher deviations.

In general, all the studied design problems (P1.1-3), with an increasing number of interface degrees of freedom n_{dof}=2, 3 to 6, respectively, were feasible. Nevertheless, a deterioration of the results compared to the previous Section 5.2 and 5.3, from 0.40% to 3.91% and a maximum of 8.18% in the current section can be observed, see Table 5.12. Thus, the higher-dimensional design space of $\mathbb{R}^8$ appears to degrade the meta model estimates in this section.

Table 5.12: Overview of the results of the Informed Decomposition (B) for design problem class (P1)

Design problem	n_{c}	n_{dof}	n_{p}	$\max\left(\frac{m_{\text{B}}-m_{\text{A}}}{m_{\text{A}}}\right)$ %	$\max(u)$ mm
(P1.1)	2	2	1(1)	0.40	≤ 0.99
(P1.2)	2	3	3(1)	3.91	≤ 1.00
(P1.3)	2	6	6(1)	8.18	≤ 1.00

6 Design problem class (P2): Towards practical application

6.1 Setup

Second, a four-component system, with a rectangular design domain Ω of width and height $w_{(i)}$=$h_{(i)}$=30 mm, and respective lengths $\boldsymbol{l}$=$[150, 100, 50, 50]^\top$ mm, is investigated in two configurations, see Fig. 6.1. The first configuration in Fig. 6.1 (a) is a planar setup, where each component i is connected in a straight way with the other components. Only rotations about the z-axis are allowed $\boldsymbol{\theta}_{(i)} = [0, 0, \gamma]^\top$. The second configuration in Fig. 6.1 (b) shows a three-dimensional robot architecture for a specific pick-and-place task with the trajectory $\boldsymbol{\theta}_{(i)}(t) = [\alpha, 0, 0]^\top$, where the joints only rotate around their local x-axis and the structural elements i are connected by different kinds of three-dimensional connectors. Both configurations of (P2), (a) and (b), are to be solved for the earlier introduced synthetic resin, see Table 5.1, and the limit on system displacement is $u_{\max}$=1 mm. The geometrical design domain Ω is again discretized with $n_{\text{ele}} = [20, 8, 8]^\top$ elements in x, y, and z-direction for each component i.

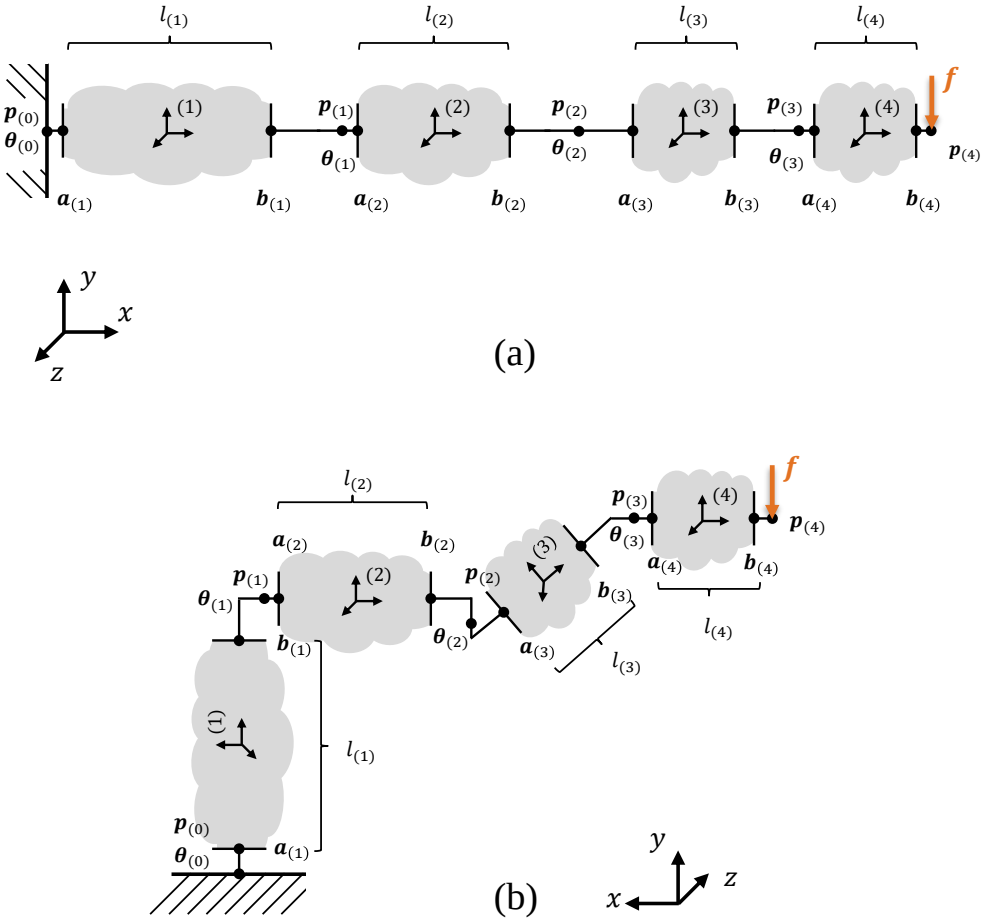


Figure 6.1: Design problem class (P2) with (a) a planar four-component system and (b) a three-dimensional four-component system, i.e., a low-cost lightweight robot. (P2.1) and (P2.2) cover the planar system, while (P2.3) investigates the three-dimensional system

The respective monolithic optimization problem (A) of (1.3) is decoupled utilizing the Informed Decomposition approach (B). As in the previous chapter, the reduced $\boldsymbol{\kappa}$-representation (4.21) with eight entries $\boldsymbol{\kappa}=\boldsymbol{\kappa}_{\mathrm{sym}} \in \mathbb{R}^{1\times 8}$ is adopted.

The main objective of this chapter is to path the way towards a practical application of (B) by highlighting some additional aspects. This is done by stepwise increasing the degrees of freedom per interface n_{dof}=2, 3, and 6 for each problem, while the Informed Decomposition (B) is investigated for

(P2.1) the general applicability of the offline database for n_{p}=1 load case in straight pose for a planar four-component system with n_{dof}=2 degrees of freedom per interface and an explicit extension of the meta models to varying geometrical dimensions, see Fig. 6.1 (a),

(P2.2) the computational time compared to two monolithic architectures (A1) and (A2), and to analytical target cascading (D), for n_{p}=1, ..., 6 simultaneously considered planar load cases for a planar four-component system with n_{dof}=3 degrees of freedom per interface, see Fig. 6.1 (a), and

(P2.3) a three-dimensional low-cost lightweight robot for n_{p}=100 static load cases considered simultaneously derived from a given dynamic trajectory $\boldsymbol{\theta}_{(i)}(t) = [\alpha, 0, 0]^{\top}$ with n_{dof}=6 degrees of freedom per interface, see Fig. 6.1 (b).

6.2 Design problem (P2.1): Varying geometrical dimensions

6.2.1 Introduction

In this section, the general applicability of the offline database is investigated, i.e., in particular, the reusability of the meta models $\hat{p}$ and $\hat{m}$ for components of different geometrical design domains $\Omega_{(i)}$. This section is also based on partial results of a similar investigation carried out in Krischer et al. (2022) and during a supervised student's project in Thomas (2022). For the design problem (P2.1), now a planar four-component system, Fig. 6.1 (a), is investigated in a straight pose with n_{dof}=2 degrees of freedom and for a payload of f=7.5 N. It consists of constant cross sections of width and height $w_{(i)}$=$h_{(i)}$=30 mm, while the lengths, $\boldsymbol{l}$=$[150, 100, 50, 50]^\top$ mm, are different. Therefore, constant scaling with λ_{xyz} for all coordinate directions, as introduced in Section 4.1.3, is not applicable.

6.2.2 Offline database

Hence, the meta models are now extended to explicit changes with respect to the x-axis

$$\hat{p}(\boldsymbol{\kappa}, \alpha_x), \tag{6.1}$$

$$\hat{m}(\boldsymbol{\kappa}, \alpha_x). \tag{6.2}$$

The sampled input design space therefore consist of $\boldsymbol{\kappa} \in \mathbb{R}^{1\times 3}$ and $\alpha_x \in \mathbb{R}^{1\times 1}$,

$$\boldsymbol{y}_A = [\boldsymbol{\kappa}_A, \alpha_{x,A}], \quad \mathbb{R}^{1\times 4}, \tag{6.3}$$

while the sample output vector remains the same

$$\boldsymbol{z}_A = [f_A, m_A], \quad \mathbb{R}^{1\times 2}. \tag{6.4}$$

The input design space $\mathbb{R}^{n_k}$ has now $n_k + n_\alpha$ dimensions, $\mathbb{R}^{n_k+n_\alpha}$, see also Fig. 6.2.

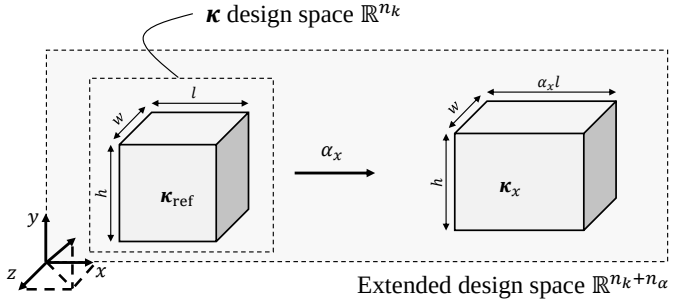


Figure 6.2: Extended design space of the meta models $\hat{p}(\boldsymbol{\kappa}, \alpha_x)$ and $\hat{m}(\boldsymbol{\kappa}, \alpha_x)$ by explicitly sampling detailed designs $\boldsymbol{x}$ for different geometrical design domains Ω with corresponding α_x

The results of the Informed Decomposition (B) are compared to the monolithic optimization approach (A). For the monolithic approach (A) no additional preparation is necessary for (P2.1). The proposed approach (B) could theoretically reuse the meta models from Section 5.2 for component i=2 of length $l_{(2)}$=100 mm. However, in this section, the goal is to train a set of meta models that can be used for the entire system with different length $l_{(i)}$. Therefore, new meta models $\hat{p}(\boldsymbol{\kappa}, \alpha_x)$ and $\hat{m}(\boldsymbol{\kappa}, \alpha_x)$ need to be created for the offline database of the Informed Decomposition (B). The general procedure for active-learning sampling remains the same as for the previous Chapter 5. The parameters for active-learning sampling can be found in Table 6.1.

Table 6.1: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P2.1)

(i) Classification sampling: $N_J = [800, 1200, 1600, \ldots, 4800]$ for $J = 1, \ldots, 10$			
Physical seed $\boldsymbol{Y}_\mathrm{p}$	Classification samples $\boldsymbol{Y}_\mathrm{C}$	Temporary samples $\boldsymbol{Y}_\mathrm{t}$	Subset of samples $\boldsymbol{Y}_\mathrm{s}$
$N_\mathrm{p} = 800$	$N_\mathrm{C} = 4800$	$N_\mathrm{t} = J^2 \frac{\left(10^6 - 10^4\right)}{10^2} + 10^4$	$N_\mathrm{s} = 400$
(ii) Regression sampling: for $A = [2726, \ldots, 5000]$			
Feasible classification samples $\boldsymbol{Y}_\mathrm{C} = 1$		Regression samples $\boldsymbol{Y}_\mathrm{R}$	
$C_\mathrm{p} = 2725$		$N_\mathrm{R} = 5000$	

The sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ is computed by the active-learning strategy with a reference length l_ref=100 mm and hence a corresponding α_x=1 value. The sample range for α_x is

$$0.48 \leq \alpha_x \leq 1.60, \tag{6.5}$$

and the results are shown in Fig. 6.3. Fig. 6.3 (i) shows the classfication sample data and Fig. 6.3 (ii) the created regression samples. In contrast to Section 5.2, the input design space is now $\mathbb{R}^4$. One can observe that within the feasible region, the volume fraction increases for higher stiffness values approaching the boundary to the infeasible region. Additionally, in accordance with the expected deformation behavior of the components, the stiffness characteristics for constant component heights h and widths w decreases for greater length l, i.e., higher α_x, and increases for lower α_x.

The shape of the physically feasible design space can cause problems for the active-learning strategy, because the algorithm is more likely to choose sample points for the low-length region, α_x<1, because of the larger design space compared to sample points of $\alpha_x \geq 1$. For instance, the $[k_{66}, \alpha_x]$-plot of Fig. 6.3 (ii) shows a lower density of sample points in the $\alpha_x \geq 1$ area.

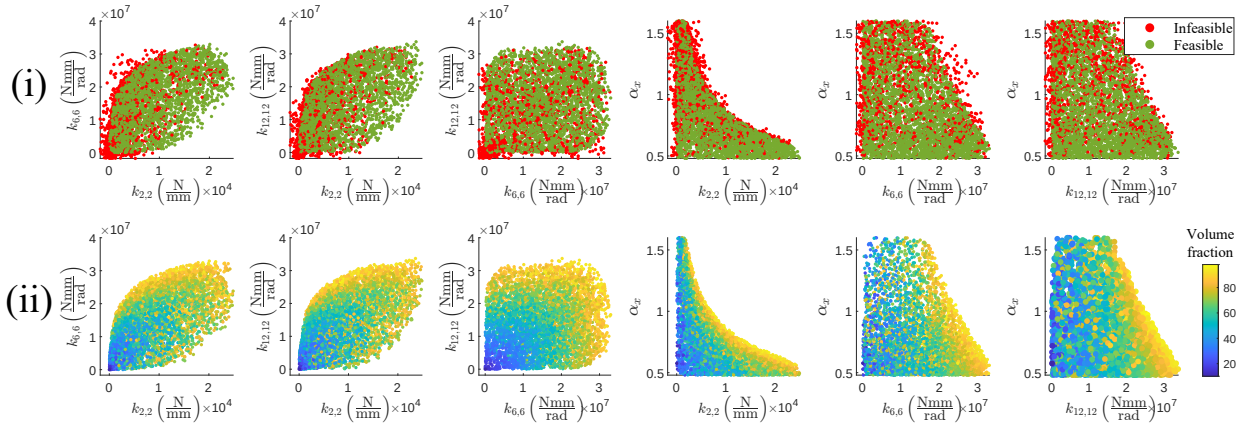


Figure 6.3: Results of the active-learning strategy for (P2.1): (i) classification samples for physical feasibility estimation and (ii) regression samples for mass estimation

The resulting sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ can then be used to train the final meta models $\hat{p}(\boldsymbol{\kappa}, \alpha_x)$ and $\hat{m}(\boldsymbol{\kappa}, \alpha_x)$ in accordance with the previous chapter. The performance measures for both estimators are listed in Table 6.2.

Table 6.2: Performance measures of $\hat{p}(\boldsymbol{\kappa}, \alpha_x)$ and $\hat{m}(\boldsymbol{\kappa}, \alpha_x)$ for (P2.1)

$\hat{p}(\boldsymbol{\kappa}, \alpha_x)$			$\hat{m}(\boldsymbol{\kappa}, \alpha_x)$	
FPR	TPR	ACC	R^2	MSE (%)
0.0433	0.916	0.927	0.992	2.92

6.2.3 System optimization

The Informed Decomposition approach (B) is applied to the design problem (P2.1) for a load case in a straight pose and compared to (A). It should be noted that despite the different lengths $\boldsymbol{l} = [150, 100, 50, 50]^\top$ mm for the components, only one feasibility estimator $\hat{p}(\boldsymbol{\kappa}, \alpha_x)$ and mass estimator $\hat{m}(\boldsymbol{\kappa}, \alpha_x)$ is utilized for the PSO system optimization (4.30) of (B). The results for the level (II) κ-representations can be seen in Table 6.3. The system optimization of (B) produces $\boldsymbol{\kappa}_{(i)}$ vectors that generally have larger values than (A) for components i=2, 3, 4. Only the first component, i=1, yields slightly lower stiffness values.

6.2.4 Component optimization

After the system optimization, the decoupled component optimizations of (B) are carried out between level (II) and (III). The resulting topologies are shown in Fig. 6.4 with the volume fractions $v_{(i)}$ and the resulting quantities on the system level (I), $\boldsymbol{z} = [m, u]$, in Table 6.3. In general, all components of approach (A) and (B) show the same features known from the previous chapter for a y-shear load case. The components i=1, 2, 3 are mainly designed against the bending load of (LC1), while the last component i=4 has the classical cantilever beam topology. The corresponding higher stiffness values for components i=2, 3, and 4 of approach (B), also lead to a higher component weight in comparison to (A), while component i=1 is accordingly slightly lighter.

In conclusion, the resulting weight m of approach (B) deviates by 4.25% to the reference mass of (A), being also physically feasible for all components. Additionally, both approaches satisfy the system stiffness requirement $u \leq u_{\max}$.

Table 6.3: Component performance (II), $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$, and the quantities on the system level (I), $\boldsymbol{z} = [m, u]$, of design problem (P2.1) for monolithic optimization (A) and the proposed Informed Decomposition (B)

Component-performance level (II): $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$						
Load case	Comp. (i)	Approach	$v_{(i)}$	$\boldsymbol{\kappa}_{(i)}$ $k_{2,2}$	$k_{6,6}$	$k_{12,12}$
			%	$\frac{\text{N}}{\text{mm}}$	$\frac{\text{N mm}}{\text{rad}}$	$\frac{\text{N mm}}{\text{rad}}$
(LC1)	(1)	(A)	78.15	878	9.65e6	9.65e6
		(B)	76.42	708	9.22e6	8.00e6
	(2)	(A)	58.68	951	8.91e6	8.55e6
		(B)	62.59	1.24e3	1.03e7	8.69e6
	(3)	(A)	35.57	972	9.61e6	7.88e6
		(B)	41.90	3.95e3	1.24e7	1.07e7
	(4)	(A)	18.46	871	2.14e6	7.20e4
		(B)	26.70	3.95e3	6.68e6	1.59e6

Quantities on the system level (I): $\boldsymbol{z} = [m, u]$					
Load case		Approach	$\sum m_{(i)}$	$\frac{(\cdot)-m_A}{m_A}$	u
			g	%	mm
(LC1)		(A)	291.7	-	1.00
		(B)	304.1	4.25	1.00

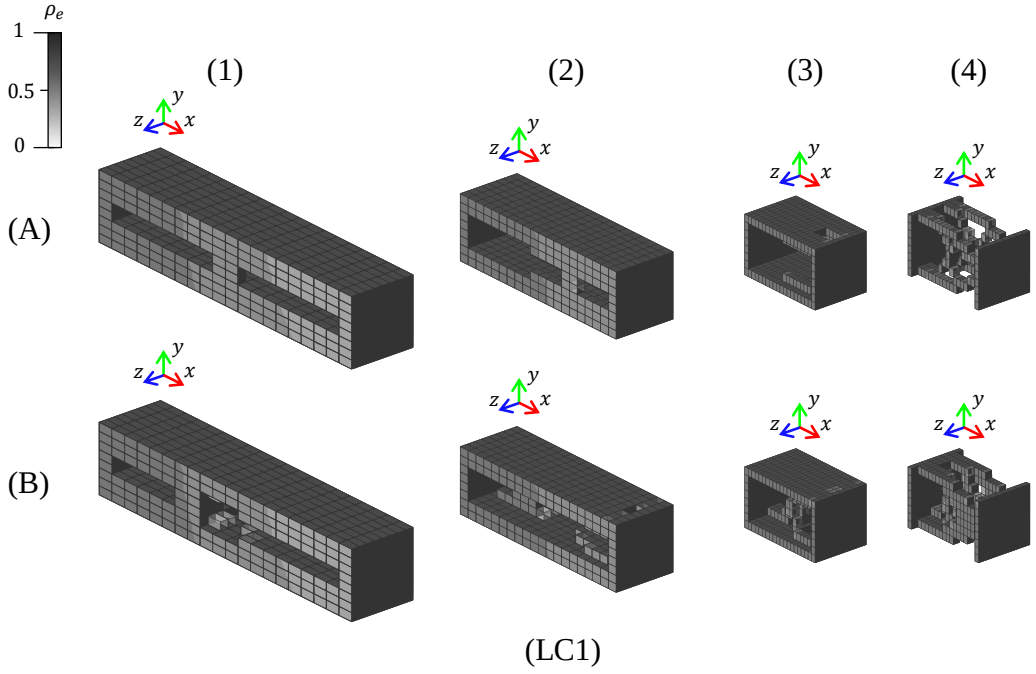


Figure 6.4: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P2.1) of the monolithic optimization (A) and Informed Decomposition (B)

6.3 Design problem (P2.2): Computational time

6.3.1 Introduction

The goal of design problem (P2.2) is to provide a detailed insight into the computational cost of the Informed Decomposition (B) and compare the results to two monolithic approaches (A1) and (A2), and ATC (D). The performance measure utilized to quantify the computational cost is the computational time t needed for executing the complete design optimization. For a design optimization, the computational time t depends on optimization driver and bottom-up mapping.

Most of the computational cost here is usually caused by solving the systems of equation of the bottom-up mappings for the component or system responses, $\boldsymbol{y}$ or $\boldsymbol{z}$, respectively, and gradients $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ and $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}$, if needed (Martins & Ning, 2022). Solvers for systems of equation of dimension $\mathbb{R}^{n\times n}$ typically have a complexity ranging from a classical Gauss-Jordan with $O(n^3)$ to $O(n^{2.376})$ of the Coppersmith-Winograd algorithm, which is however rarely used in practice (Pan, 1987). The same holds for multiplication of two matrices of dimension $\mathbb{R}^{n\times n}$ (Cormen et al., 2022).

The total complexity for a bottom-up mapping of a general multi-component system depends on the number of

(1) total degrees of freedom n_d of each component i,
(2) components n_c, and
(3) load cases n_p considered simultaneously, if $\boldsymbol{\theta}_c \neq \boldsymbol{\theta}_j$.

In order to investigate the computational time of the architectures (A1-2), (B), and (D), the complexity of design problem (P2.2), with n_dof=3 degrees of freedom per interface and a constant number of total degrees of freedom n_d of each component, is slowly increased by increasing the number of

(P2.2.1) i=1, 2, 3, and 4 components for load case (LC1), see Fig. 6.5 (a), and
(P2.2.2) c=1, ..., 6 load cases for a four-component system, see Fig. 6.5 (b) and Table 6.4.

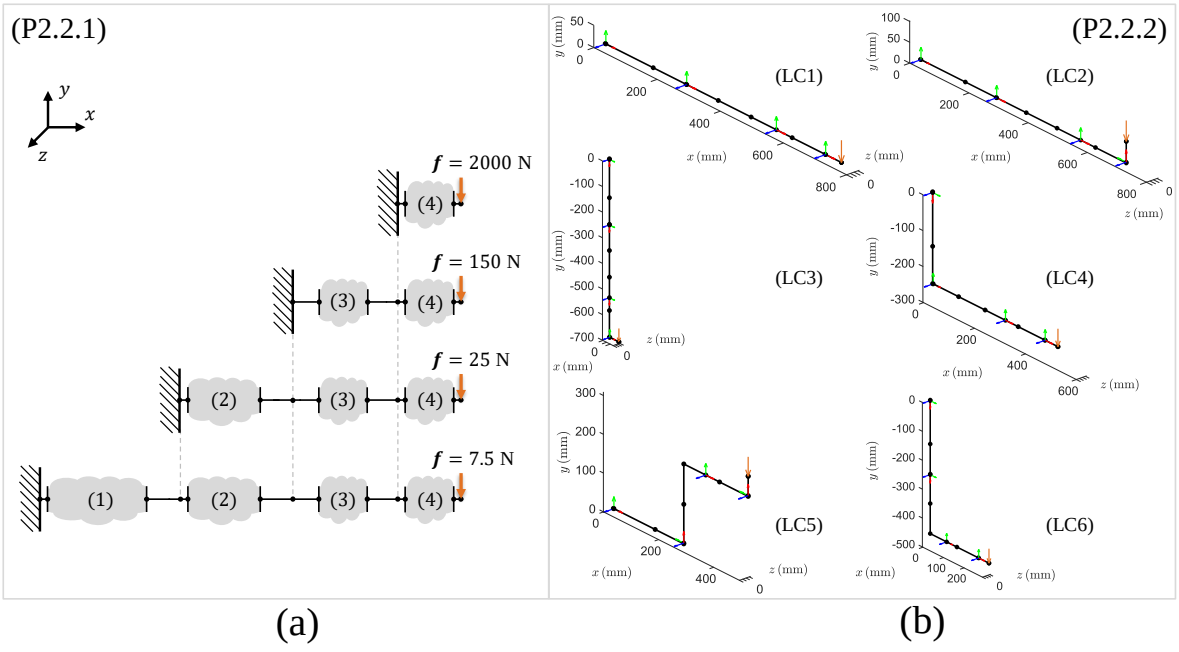


Figure 6.5: Design problem (P2.2) with a planar four-component system that (a) is set up starting from i=1, 2, 3 to 4 components and (b) the full four-component system with increasing number of c=1, ..., 6 load cases (LC1-6) that are considered simultaneously

Table 6.4: Design problem (P2.2): Input angles $\boldsymbol{\theta}$, forces $\boldsymbol{f}$, joint positions $\boldsymbol{p}$, and interface positions $\boldsymbol{a}$ and $\boldsymbol{b}$ for the n_p=6 load cases (LC1-6)

Load case	$\boldsymbol{\theta}_{(i)}$ rad	$\boldsymbol{f}$ N	$\boldsymbol{p}_{(0)}$ mm	$\boldsymbol{a}_{(1)}, \boldsymbol{b}_{(1)}$ mm	$\boldsymbol{p}_{(1)}$ mm	$\boldsymbol{a}_{(2)}, \boldsymbol{b}_{(2)}$ mm	$\boldsymbol{p}_{(2)}$ mm	$\boldsymbol{a}_{(3)}, \boldsymbol{b}_{(3)}$ mm	$\boldsymbol{p}_{(3)}$ mm	$\boldsymbol{a}_{(4)}, \boldsymbol{b}_{(4)}$ mm	$\boldsymbol{p}_{(4)}$ mm
(LC1)	$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-7.5\end{bmatrix}$									
(LC2)	$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-9\end{bmatrix}$									
(LC3)	$\begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-80\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}2\\0\\0\end{bmatrix}\begin{bmatrix}152\\0\\0\end{bmatrix}$	$\begin{bmatrix}256\\0\\0\end{bmatrix}$	$\begin{bmatrix}258\\0\\0\end{bmatrix}\begin{bmatrix}358\\0\\0\end{bmatrix}$	$\begin{bmatrix}462\\0\\0\end{bmatrix}$	$\begin{bmatrix}542\\0\\0\end{bmatrix}\begin{bmatrix}592\\0\\0\end{bmatrix}$	$\begin{bmatrix}696\\0\\0\end{bmatrix}$	$\begin{bmatrix}698\\0\\0\end{bmatrix}\begin{bmatrix}748\\0\\0\end{bmatrix}$	$\begin{bmatrix}750\\0\\0\end{bmatrix}$
(LC4)	$\begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-8\end{bmatrix}$									
(LC5)	$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\\-15\end{bmatrix}$									
(LC6)	$\begin{bmatrix}0\\0\\-\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\\frac{\pi}{2}\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\-18\end{bmatrix}$									

The monolithic optimization (A) is carried out considering n_p load cases simultaneously, for the two architectures (A1) and (A2). So does the system optimization of (B), while the component optimization of (B) remains unaffected by the number of components n_c or number of load cases n_p.

Lastly, the system optimization of the ATC (D) for the given design problem class (P2.2) is

$$\begin{aligned} &\min_{\boldsymbol{\kappa}^t_{(i)}, m^t_{(i)}} \sum_{i=1}^{n_c} m^t_{(i)} + \sum_{i=1}^{n_c} \left(p_{\mathrm{m},(i)} \frac{m_{(i)} - m^t_{(i)}}{m_{\mathrm{ub},(i)} - m_{\mathrm{lb},(i)}} \right)^2 + \sum_{i=1}^{n_c} \sum_{j=1}^{4} \left(p_{\mathrm{k},(i),j} \frac{\kappa_{(i),j} - \kappa^t_{(i),j}}{\kappa_{\mathrm{ub},(i),j} - \kappa_{\mathrm{lb},(i),j}} \right)^2, \\ \text{s. t.:} \quad & u_c(\boldsymbol{\kappa}^t_{(i)}) - u_{\max} \leq 0, \qquad \text{for } c{=}1, ..., n_{\mathrm{p}}, \\ & \boldsymbol{\phi}^\top_{(r,i)} \boldsymbol{K}_{(i)} \boldsymbol{\phi}_{(r,i)} \geq 0, \qquad \text{for } r{=}1, ..., n_\phi \text{ and for } i{=}1, ..., n_{\mathrm{c}}, \end{aligned} \tag{6.6}$$

where $\boldsymbol{K}_{(i)}{=}\boldsymbol{\Phi}(\boldsymbol{\kappa}^t_{(i)})$ and the component optimization is

$$\begin{aligned} &\min_{\boldsymbol{x}_{(i)}} m_{(i)}(\boldsymbol{x}_{(i)}), \\ \text{s. t.:} \quad & -\epsilon_{\mathrm{k}} \leq \frac{\kappa_{(i),j} - \kappa^t_{(i),j}}{\kappa_{\mathrm{ub},(i),j} - \kappa_{\mathrm{lb},(i),j}} \leq \epsilon_{\mathrm{k}}, \quad \text{for } j = 1, ..., 4, \\ & -\epsilon_{\mathrm{m}} \leq \frac{m_{(i)} - m^t_{(i)}}{m_{\mathrm{ub},(i)} - m_{\mathrm{lb},(i)}} \leq \epsilon_{\mathrm{m}}. \end{aligned} \tag{6.7}$$

The target values $[\boldsymbol{\kappa}^t, \boldsymbol{m}^t]$ are the design variables of the system optimization (6.6) and $[\boldsymbol{\kappa}, \boldsymbol{m}]$ are the actual responses computed by the detailed design variables $\boldsymbol{x}_{(i)}$ of the component optimizations (6.7). Since no meta models are utilized for the system optimization, an additional constraint on positive semi-definiteness for the stiffness matrices $\boldsymbol{K}_{(i)} \in \mathbb{R}^{6\times6}$ is added.

The original component optimization (3.5) of (D) contains a penalty function as in the system optimization (6.6). However, the MMA optimization algorithm utilized to solve the component optimization is sensitive with respect to the magnitude of the objective function values. The penalty scheme may cause problems in terms of convergence behavior. Therefore, the original formulation is adapted by reformulating the penalty as two-sided inequality constraints where ϵ for the stiffness and mass values is continuously reduced during the outer loop. This is similar to the effect of increasing the weights p in a penalty function formulation of the original ATC component optimization.

In the following, a rough estimation on the computational cost of the four architectures (A1-2), (B), and (D) for design problem (P2.2) is conducted. For this, a complexity of $O(n^3)$ is assumed for all main computations to illustrate the influence of the number of components n_c and load cases n_p for a constant number of total degrees of freedom n_d per component, while other constants are neglected.

6.3.2 Cost estimation

Normal bottom-up mapping (A1)

In industry, a FEM model is typically built as a black-box function using a commercial software without any explicit hierarchy levels (I, II, and III) that were introduced in Section 1.2. We will refer to the monolithic architecture that is utilizing such a bottom-up mapping as a normal monolithic optimization (A1). The approach (A1) resembles the classical situation for an optimization task in industry. Here, one would simply set up each structural element of component i first and then connect them to each other by, e.g., applying a rigid body element to both sides $\boldsymbol{A}$ and $\boldsymbol{B}$ of $\boldsymbol{K}_\mathrm{d} \in \mathbb{R}^{n_\mathrm{d} \times n_\mathrm{d}}$ with $n_\mathrm{d} = n_\mathrm{m} + n_\mathrm{int} n_\mathrm{dof}$. The degrees of freedom of the finite element mesh are n_m, n_dof are the degrees of freedom of the interfaces, and n_int is the number of interfaces, see Fig. 6.6 (a).

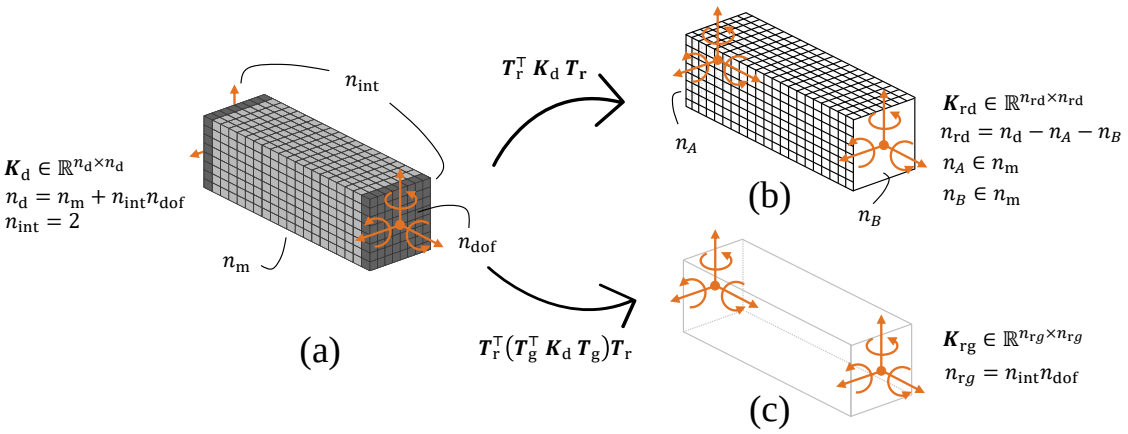


Figure 6.6: Modeling process starting from (a) $\boldsymbol{K}_d$ containing all degrees of freedom of the structural element and reduction to either (b) $\boldsymbol{K}_{rd}$ of (A1) from the normal bottom-up mapping, or (c) $\boldsymbol{K}_{rg}$ of (A2, B, D) for the hierarchical bottom-up mappings

Therefore, the detailed stiffness matrix, see Fig. 6.6 (b), can be built independently of the number of load cases n_p for each of the n_c components

$$\boldsymbol{K}_\mathrm{rd} = \boldsymbol{T}_\mathrm{r}^\top \boldsymbol{K}_\mathrm{d} \boldsymbol{T}_\mathrm{r}, \qquad \boldsymbol{K}_\mathrm{rd} \in \mathbb{R}^{n_\mathrm{rd} \times n_\mathrm{rd}}, \tag{6.8}$$

where the degrees of freedom n_A and n_B are subtracted from the model, i.e., n_rd=n_d−n_A−n_B. Note that unlike the reduction procedure of Section 4.1, no Guyan reduction $\boldsymbol{T}_\mathrm{g}$ is applied to the interior nodes of the component and the computation of $\boldsymbol{T}_\mathrm{r}$ (2.17) and $\boldsymbol{K}_\mathrm{rd}$ (6.8) is neglected in the following estimation.

The overall FEM equation, involving the system stiffness matrix $\boldsymbol{K}_{\mathrm{rd,s}}$, is then

$$\begin{aligned} \boldsymbol{K}_{\mathrm{rd,s}}\boldsymbol{d}_{\mathrm{rd,s}} = &\quad \boldsymbol{f}_{\mathrm{rd,s}}, && \boldsymbol{K}_{\mathrm{rd,s}} \in \mathbb{R}^{n_{\mathrm{rd,s}}\times n_{\mathrm{rd,s}}}, \\ n_{\mathrm{rd,s}} = &\quad n_{\mathrm{c}}n_{\mathrm{rd}} - n_{\mathrm{c}}n_{\mathrm{dof}} \approx n_{\mathrm{c}}n_{\mathrm{rd}}, && \text{for } n_{\mathrm{rd}} >> n_{\mathrm{dof}}, \end{aligned} \tag{6.9}$$

where the assembly operations for $\boldsymbol{K}_{\mathrm{rd,s}}$, consisting of the joint condensation matrices $\boldsymbol{T}_{p,(i-1,i)}$, the rotation matrices $\boldsymbol{R}_{(i)}$, and the assembly operator $\mathrm{A}_{i=1}^{n_{\mathrm{c}}}$ involving the detailed stiffness matrices $\boldsymbol{K}_{rd,(i)} \in \mathbb{R}^{n_{\mathrm{rd}}\times n_{\mathrm{rd}}}$, are also neglected for this computational cost estimation.

For the serial assembly process, the first interface of the system is clamped and the overlapping interfaces from the subsequent components are subtracted. However, the degrees of freedom n_{rd} are much larger than n_{dof} of the interfaces. Hence, the size of the system is approximately correlated with the number of components n_{c} and their degrees of freedom n_{rd}, i.e., $n_{\mathrm{rd,s}} \approx n_{\mathrm{c}}n_{\mathrm{rd}}$.

The system is then solved at once using (6.9). If a new load case j with a different pose $\boldsymbol{\theta}_j \neq \boldsymbol{\theta}_c$ is introduced, a new bottom-up mapping must be created involving the inversion of $\boldsymbol{K}_{\mathrm{rd,s}} \in \mathbb{R}^{n_{\mathrm{rd,s}}\times n_{\mathrm{rd,s}}}$. For more than one load case, $n_{\mathrm{p}}>1$, the normal monolithic optimization (A1) performs this expensive bottom-up computation n_{p} times in serial for each load case $c=1, \ldots, n_{\mathrm{p}}$, see Fig. 6.7 (A1). Hence, the total computational complexity of (6.9) for a Gauss-Jordan solver is

$$O(n_{\mathrm{norm}}^3) = \quad O\left(n_{\mathrm{p}}\left[n_{\mathrm{rd,s}}\right]^3\right) \approx O\left(n_{\mathrm{p}}\left[n_{\mathrm{c}}n_{\mathrm{rd}}\right]^3\right). \tag{6.10}$$

If for $n_{\mathrm{rd}}=n_{\mathrm{d}}-n_A-n_B$, it is assumed that the total number of degrees of freedom n_{d} is much larger than those of the left n_A and the right n_B, the computational complexity of the entire normal bottom-up mapping, for $n_{\mathrm{c}}=4$ and $n_{\mathrm{p}}=6$, can be expressed with respect to n_{d}

$$O(n_{\mathrm{norm}}^3) \approx \quad O\left(n_{\mathrm{p}}\left[n_{\mathrm{c}}n_{\mathrm{d}}\right]^3\right) = O\left(384\, n_{\mathrm{d}}^3\right), \qquad \text{for } n_{\mathrm{rd}} \approx n_{\mathrm{d}}. \tag{6.11}$$

Thus, the computation of the bottom-up mapping of (A1) for (P2.2) is up to 384 times higher than the computation of one load case for one component, i.e., $n_{\mathrm{p}}=1$ and $n_{\mathrm{c}}=1$. This may make an actual monolithic optimization (A1) for more complex design problems prohibitively expensive.

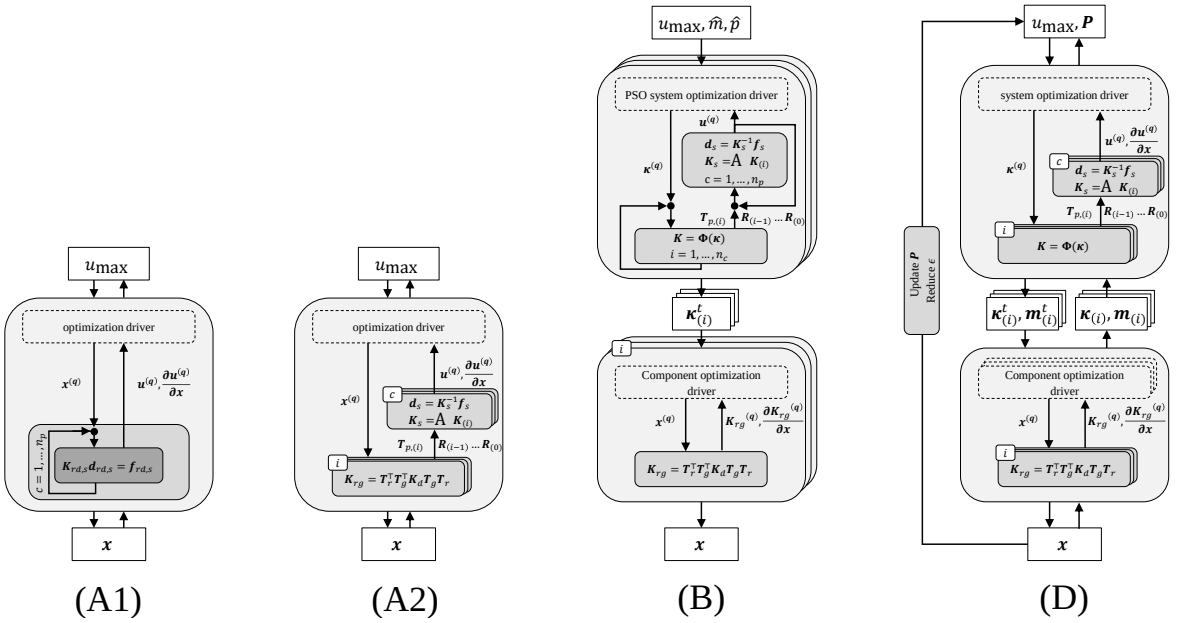


Figure 6.7: Flowchart of the two monolithic optimizations (A1-2), Informed Decomposition (B) and analytical target cascading (D)

Hierarchical bottom-up mappings (A2, B, D)

However, the earlier introduced hierarchy levels (I, II, and III) can be used to establish a hierarchical and also decomposed formulation consisting of two main bottom-up mappings instead of one monolithic one, which makes the computations approximately independent of the number of components n_c and load cases n_p.

First, between level (III) and (II), the interface stiffness matrix $\boldsymbol{K}_{rg} \in \mathbb{R}^{n_{rg} \times n_{rg}}$, for $n_{rg}=2n_{dof}$, is computed from $\boldsymbol{K}_d \in \mathbb{R}^{n_d \times n_d}$, see also Fig. 6.6 (c),

$$\boldsymbol{K}_{rg} = \boldsymbol{T}_r^\top \boldsymbol{T}_g^\top \boldsymbol{K}_d \boldsymbol{T}_g \boldsymbol{T}_r, \qquad \boldsymbol{K}_d \in \mathbb{R}^{n_d \times n_d} \rightarrow \boldsymbol{K}_{rg} \in \mathbb{R}^{n_{rg} \times n_{rg}}. \tag{6.12}$$

The computational complexity of $\boldsymbol{T}_g$ (2.10) and $\boldsymbol{T}_r$ (2.17) correlates to the number of slave degrees of freedom s with $O(n^3)$. The same holds approximately for the dimensions of the four sequential matrix multiplications to reduce the detailed stiffness matrix $\boldsymbol{K}_d$ from $\mathbb{R}^{n_d \times n_d}$ to $\mathbb{R}^{n_{rg} \times n_{rg}}$. However, $\boldsymbol{K}_{rg}$ is computed independently of the number of load cases n_p and in parallel for n_c components.

For a rough estimation of the bottom-up mapping (6.12), the computational complexity is therefore related to the highest involved number of degrees of freedom n_d

$$O(n_{d\rightarrow rg}^3) \approx O\left(n_d^3\right). \tag{6.13}$$

Second, between level (II) and (I) for each component i, the condensation and rotational matrices $\boldsymbol{T}_{p,(i-1,i)}$ and $\boldsymbol{R}_{(i)}$ are applied to $\boldsymbol{K}_{rg,(i)}$ and the final assembly process $\mathrm{A}_{i=1}^{n_c}$ is carried out. The bottom-up mapping between level (II) and (I), for n_c components and $n_{rg}=2n_{dof}$, is then

$$\boldsymbol{K}_s \boldsymbol{d}_s = \quad \boldsymbol{f}_s, \qquad \boldsymbol{K}_s \in \mathbb{R}^{n_s \times n_s} \text{ with } n_s = n_c n_{rg} - n_c n_{dof} = n_c n_{dof}. \tag{6.14}$$

The number of load cases n_p now only affects the lower dimensional system stiffness matrix $\boldsymbol{K}_s \in \mathbb{R}^{n_s \times n_s}$ based on $\boldsymbol{K}_{rg}$ instead of the high-dimensional detailed system stiffness matrix $\boldsymbol{K}_{rd,s} \in \mathbb{R}^{n_{rd,s} \times n_{rd,s}}$ based on $\boldsymbol{K}_{rd}$ of (A1), see also Fig. 6.6 (b)-(c).

Neglecting the assembly process, the computational complexity for the bottom-up mapping (6.14) between level (II) and (I) for n_p load cases is then

$$O(n_p n_s^3) = O\left(n_p \left[n_c n_{dof}\right]^3\right). \tag{6.15}$$

The degrees of freedom n_s of the bottom-up mapping (6.14) between level (II) and (I) are negligible compared to $n_{d\rightarrow rg}$ of the detailed bottom-up mapping (6.12) between (III) and (II), i.e.,

$$n_{d\rightarrow rg} >> n_s. \tag{6.16}$$

Therefore, the total computational complexity of the hierarchical bottom up mappings is approximately equal to (6.13). This means, that the computational cost $O(n_{hier}^3)$ is independent of the number of components n_c and load cases n_p, if the necessary resources for parallelization are available

$$O\left(n_{hier}^3\right) \approx O\left(n_d^3\right). \tag{6.17}$$

This type of hierarchical bottom-up mappings is used for the second monolithic architecture (A2) as well as for the two distributed architectures of Informed Decomposition (B) and ATC (D), see Fig. 6.7. In contrast to (A2), which uses the two decomposed hierarchical bottom-up mappings for a single optimization problem in parallel, the distributed architectures have a bottom-up mapping for each optimization subproblem. Consequently, the system optimization uses (6.14), while the component optimization uses (6.12).

Full matrix vs. sparse matrix operations

For sparse matrices, such as assembled stiffness matrices $\boldsymbol{K}$, the solution process can be greatly accelerated by a special sparse implementation. Here, the computational complexity depends on $nnz(\boldsymbol{K})$, the number of nonzero elements in $\boldsymbol{K}$ (Gilbert et al., 1992). This effect is particularly important for the normal bottom-up mapping of (A1), since $\boldsymbol{K}_{\mathrm{rd,s}}$ has the largest dimensions $n_{\mathrm{rd,s}}$ that scale with $O(n^3)$ for a Gauss-Jordan solver. Table 6.5 shows a comparison between the normal bottom-up mapping of (A1) and the corresponding hierarchical bottom-up mappings of (A2), (B), and (D) for full and sparse matrix operations for n_{c}=4 components and n_{p}=6 load cases. Each operation was carried out n_t=20 times and the average was taken.

Table 6.5: Computational time of the normal bottom-up mapping (A1) and the hierarchical bottom-up mappings of (A2),(B), and (D) for full and sparse matrix implementation

Bottom-up mapping			
full		sparse	
norm.	hier.	norm.	hier.
t_{norm} = 400 s	t_{hier} = 1.39 s	t_{norm} = 2.12 s	t_{hier} = 0.402 s
$\frac{t_{\mathrm{norm}}}{t_{\mathrm{hier}}} = 287$		$\frac{t_{\mathrm{norm}}}{t_{\mathrm{hier}}} = 5.27$	

It can be observed that for full matrices the time ratio between the normal approach and the hierarchical one is with $\frac{t_{\mathrm{norm}}}{t_{\mathrm{hier}}} = 287$ close to the estimated 384 of (6.11). A reason for the slightly lower value could be that the internal solver of MATLAB is quicker than the estimated complexity of $O(n^3)$. With the usage of sparse operations, the time difference decreases drastically but is still showing a significant ratio of $\frac{t_{\mathrm{norm}}}{t_{\mathrm{hier}}} = 5.27$.

Yet, the computational cost necessary to solve the main bottom-up mappings for the responses $\boldsymbol{z}$ and $\boldsymbol{y}$ is not the only factor contributing to computational time. Additionally, some internal assembly computations are needed and for parallel activities also a coordination between the different computations takes place. For gradient-based algorithms, the gradient computations within the bottom-up mappings also take additional time but are usually smaller than the response computations.

Optimization driver

The optimization driver is the second source of computational cost within design optimization. For (P2.2), the optimization driver solves different optimization problems for the approaches (A1-2), (B), and (D).

Both monolithic optimizations (A1-2) solve the original monolithic optimization problem (1.3) with the respective detailed design variables of the high-dimensional level (III)

$$\boldsymbol{x}=[\boldsymbol{x}_{(1)},...,\boldsymbol{x}_{(n_{\mathrm{c}})}]. \tag{6.18}$$

In contrast, the system optimization for (B) and (D) works on the component-performance level (II)

$$\boldsymbol{\kappa}=[\boldsymbol{\kappa}_{(1)},...,\boldsymbol{\kappa}_{(n_{\mathrm{c}})}], \tag{6.19}$$

and the component optimizations only deal with their local variables of level (III)

$$\boldsymbol{x}_{(i)}. \tag{6.20}$$

In the following, the actual computational time t needed for the design problems (P2.2.1) and (P2.2.2) is computed. A similar investigation was performed in the supervised student's project of Rinner (2022), but for different design problems and with a comparison only between monolithic optimization (A) and the Informed Decomposition (B).

6.3.3 Cost investigation

Total computational time t and average computational time $\bar{t}$

As mentioned earlier, the main performance measure for computational cost within this section is the computational time t needed for executing the complete design optimization. The number of iterations q=1, ..., n_{iter} together with the computational time per iteration $t^{(q)}$ of the utilized bottom-up mappings and the algorithm-specific computations of the optimization driver, result in the computational time t of the entire optimization. Since (B) is based on the idea of an offline database, the sampling and training time t_{offline} for the meta models is not considered during this time investigation.

Despite the computational time t, also the number of iterations n_{iter} are taken into account by computing the average computational time per iteration $\bar{t}$. For the following investigation, each optimization is carried out n_t=20 times to compensate for performance variations during the optimization and the mean of all runs is taken

$$t = \frac{1}{n_t} \sum_{i=1}^{n_t} t_i, \tag{6.21}$$

$$\bar{t} = \frac{t}{n_{\text{iter}}}. \tag{6.22}$$

For the different (sub-)problems of (A1), (A2), (B), and (D), different optimization algorithms are utilized. For a fair comparison, the same convergence criteria for all optimizations concerned with the expensive level (III) was used. The system optimizations between level (I) and (II) have different criteria since (B) is solved by a PSO and (D) by an interior point method utilizing the fmincon algorithm of MATLAB, see Table 6.6.

Table 6.6: Optimization settings for (A1-2), (B), and (D)

	(A1) (A2)	(B) (D)	(B)	(D)
Level	(I)-(III)	(II)-(III)	(I)-(II)	
Algorithm	MMA		PSO	fmincon
Convergence criteria	$\lvert \boldsymbol{x}^{(q+1)} - \boldsymbol{x}^{(q)} \rvert_\infty \leq 1e{-}3$			$\lvert \boldsymbol{x}^{(q+1)} - \boldsymbol{x}^{(q)} \rvert_\infty \leq 1e{-}10$
			$\frac{\lvert f(\boldsymbol{x}^{(q)}) - f(\boldsymbol{x}^{(q+1)}) \rvert}{1+\lvert f(\boldsymbol{x}^{(q)}) \rvert} \leq 1e{-}2$	$\frac{\lvert f(\boldsymbol{x}^{(q)}) - f(\boldsymbol{x}^{(q+1)}) \rvert}{1+\lvert f(\boldsymbol{x}^{(q)}) \rvert} \leq 1e{-}6$

All optimizations were performed on the same computer, with all files stored on the local hard disk. The resources used are listed in Table 6.7.

Table 6.7: Computational resources used for the computational time comparison of (P2.2)

Processor	Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz
Memory	16 GB
Software	MATLAB R2021a

Offline Database

For (P2.2), the meta models of Section 5.3 for component i=2 with l=100 mm can be reused and therefore only two more meta models with input $\boldsymbol{\kappa} \in \mathbb{R}^{1\times 4}$ for l=50 mm and l=150 mm need to be trained

$$\hat{p}_{l=50}(\boldsymbol{\kappa}), \quad \hat{p}_{l=150}(\boldsymbol{\kappa}), \tag{6.23}$$

$$\hat{m}_{l=50}(\boldsymbol{\kappa}), \quad \hat{m}_{l=150}(\boldsymbol{\kappa}). \tag{6.24}$$

The parameters for the active-learning sampling can be found in Table 6.8. For classification sampling (i), the same parameters are used for l=[50, 100] mm as for l=100 mm of Section 5.3.

Table 6.8: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P2.2) for l=50 mm and l=150 mm

(i) Classification sampling: $N_J = [600, 1200, ..., 6600]$ for $J = 1, ..., 10$			
Physical seed $\boldsymbol{Y}_\mathrm{p}$	Classification samples $\boldsymbol{Y}_\mathrm{C}$	Temporary samples $\boldsymbol{Y}_\mathrm{t}$	Subset of samples $\boldsymbol{Y}_\mathrm{s}$
$N_\mathrm{p} = 600$	$N_\mathrm{C} = 6600$	$N_\mathrm{t} = J^4 \frac{\left(10^6 - 10^5\right)}{10^4} + 10^5$	$N_\mathrm{s} = 600$
(ii) Regression Sampling: for $A = [C_\mathrm{p} + 1, ..., 8000]$			
Feasible classification samples $\boldsymbol{Y}_\mathrm{C} = 1$		Regression samples $\boldsymbol{Y}_\mathrm{R}$	
l = 50 mm	l = 150 mm	l = 50 mm	l = 150 mm
$C_\mathrm{p} = 4338$	$C_\mathrm{p} = 4272$	$N_\mathrm{R} = 8000$	

The total sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ calculated with the active-learning strategy for l=50 mm and l=150 mm are shown in Fig. 6.8 (a) and (b), respectively, while (i) shows the classification sample data and (ii) the regression sample data. The general shape is similar to the results of Fig. 5.8 from Section 5.3. However, the magnitude of the stiffness entries of $\boldsymbol{\kappa}_{(i)}$ differs significantly due to the different component lengths $l_{(i)}$.

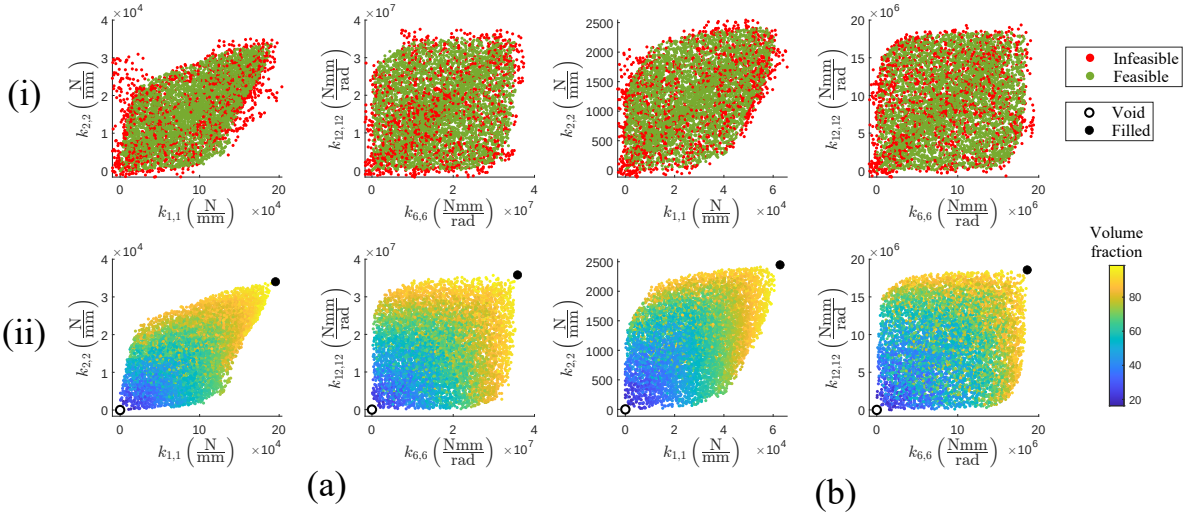


Figure 6.8: Results of the active-learning strategy for (P2.2) for (a) l=50 mm and (b) l=150 mm. The first row are the (i) classification samples for physical feasibility estimation and the second row are the (ii) regression samples for mass estimation

The sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ can then be used to train the final meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ as in Section 5.2. The performance measures for both estimators are listed in Table 6.9.

Table 6.9: Performance measures of $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for (P2.2)

l=50 mm					l=150 mm				
$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$		$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$	
FPR	TPR	ACC	R^2	MSE (%)	FPR	TPR	ACC	R^2	MSE (%)
0.0452	0.937	0.942	0.994	1.73	0.0292	0.925	0.937	0.992	2.50

(P2.2.1) Increasing number of components n_c

For the first investigation (P2.2.1), the multi-component system is slowly assembled by increasing the number of components from $n_c = 1, 2, 3$ to 4, see Fig. 6.5 (a). The results of the computational cost for the approaches (A1-2), (B), and (D) are shown in Fig. 6.9. In Fig. 6.9 (a), the computational time t of all approaches can be seen. Since applying distributed architectures to a one-component system is not reasonable, only the monolithic approaches (A1-2) were performed for this case. The first thing to notice is that the computational time $t_{(D)}$ of approach (D) is significantly larger than those of all other approaches.

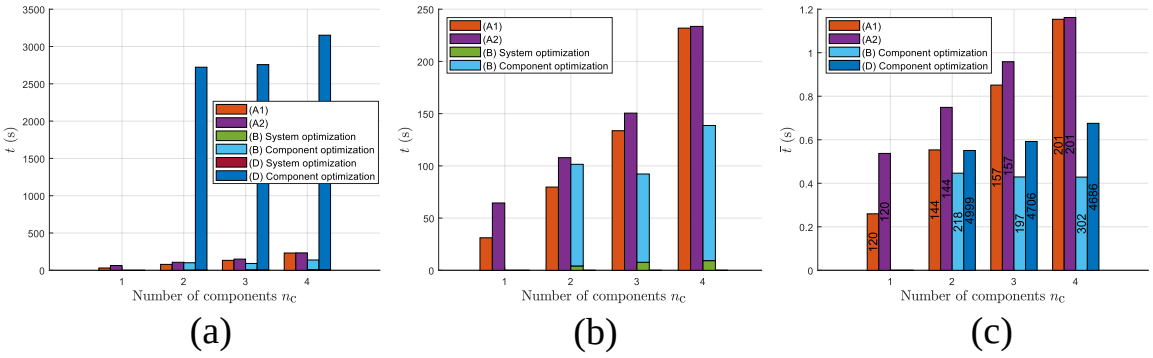


Figure 6.9: Time comparison for (P2.2.1): (a) and (b) total computational time t, (c) computational time per iteration $\bar{t}$

Therefore, Fig. 6.9 (b) shows the comparison on a smaller time scale neglecting approach (D). Here, the computational time $t_{(A1)}$ for approach (A1) increases as the number of components n_c increases. In comparison, approach (A2) is slightly slower and shows the same trend as the number of components increases. Based on the estimation (6.17), independence of the number of components n_c was expected for (A2) and hence also a lower computational time $t_{(A2)}$. This contradiction can be explained by the internal coordination required for parallelization, which seems to be higher for the given given problem than the advantage of running the bottom-up mappings of size n_d in parallel. In particular, for n_c=1, the normal approach (A1) is shown to have a lower computational time $t_{(A1)}$ compared to $t_{(A2)}$, demonstrating the slower data management.

In contrast, the Informed Decomposition (B), for n_c>2, is the fastest of all approaches, exploiting the potential of parallelization especially for n_c=4. Since no internal coordination is required for the decoupled component optimizations, the speedup is clearly visible. Moreover, one can observe that the assumption (6.16) of negligible system optimizations was reasonable, since system optimization accounts for only a small percentage of the computational time $t_{(B)}$. However, there are significant differences in the computational time $t_{(B)}$ for each component configuration.

To get a more detailed insight, Fig. 6.9 (c) shows the average computational time $\bar{t}$ and the needed iterations n_{iter}. For approach (B), the average computational time $\bar{t}_{(B)}$ is almost the same in all cases, showing that only the number of iterations $n_{iter} = [218, 197, 302]$ caused the differences. Since the component optimization is almost the same for (B) and (D), it can be observed that the average computational time $\bar{t}_{(D)}$ is also only slightly higher. The difference in Fig. 6.9 (a) can therefore be explained by internal coordination within the inner loop of the distributed optimization of (D). This coordination overhead lead to n_{iter}>4000 iterations for all component configurations n_c=1, 2, 3, 4. Further, approach (A1) shows again a slightly lower $\bar{t}$ compared to (A2). This is plausible since (A1) and (A2) solve the same optimization problem and therefore have the same number of iterations n_{iter} for each design problem.

Moreover, the resulting quantities of interest are shown in Table 6.10. Fig. 6.10 shows the final topologies for load case (LC1) and n_c=4 components for (A), (B), and (D). All approaches are able to satisfy the system requirement $u \leq u_{max}$. However, while approach (B) has a maximum mass deviation of 5.55%, approach (D) not only has a higher computational time $t_{(D)}$, but also problems finding a design close to the reference design of (A) with a maximum deviation of 24.0%. The identical results for approach (A1) and (A2) also prove that different bottom-up mappings do not affect the results, but only the computational time t.

Table 6.10: Quantities of interest of (P2.2.1) for monolithic optimizations (A1-2), Informed Decomposition (B), and analytical target cascading (D)

Number of components n_c	Approach	$\sum m_{(i)}$ g	$\frac{(\cdot)-m_A}{m_A}$ %	u mm	Number of components n_c	Approach	$\sum m_{(i)}$ g	$\frac{(\cdot)-m_A}{m_A}$ %	u mm
1	(A1)	18.02	-	1.00	2	(A1)	35.79	-	1.00
	(A2)	18.02	-	1.00		(A2)	35.79	-	1.00
	(B)	-	-	-		(B)	37.01	3.41	0.99
	(D)	-	-	-		(D)	44.38	24.0	0.95
3	(A1)	115.1	-	1.00	4	(A1)	291.7	-	1.00
	(A2)	115.1	-	1.00		(A2)	291.7	-	1.00
	(B)	120.4	4.55	1.00		(B)	307.9	5.55	1.00
	(D)	129.5	12.4	1.00		(D)	332.9	14.1	1.00

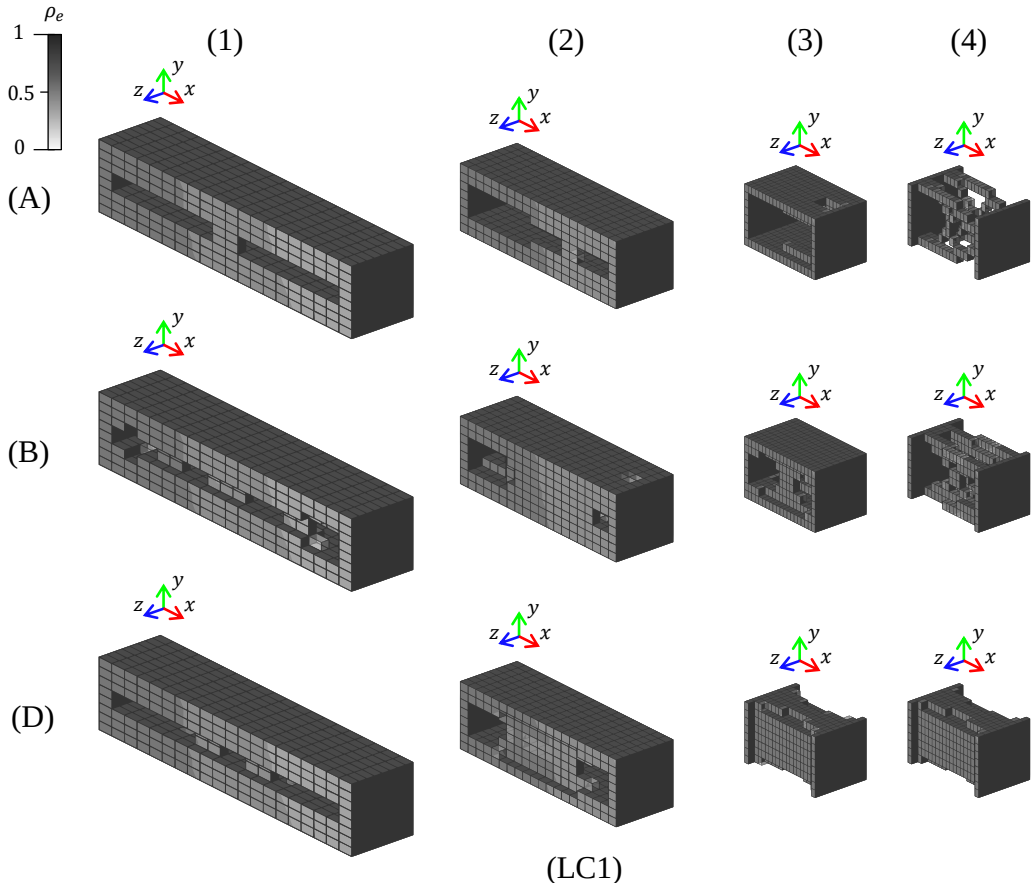


Figure 6.10: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P2.2.1) of the monolithic optimization (A1-2), Informed Decomposition (B), and analytical target cascading (D) for n_c=4 components

(P2.2.2) Increasing number of load cases n_p

Due to the large coordination overhead of approach (D), it is neglected for the second investigation. For (P2.2.2), the number of load cases n_p is increased for the full four-component system, see Fig. 6.5 (b). Note that unlike Chapter 5, the load cases are all part of the same optimization problem and hence are considered simultaneously. Moreover, the load cases are arranged in such a way that load case (LC1) is the dominant load case, i.e., $\boldsymbol{x}^*$ for (LC1) is feasible for all other load case configurations (LC1-LC6), and thus is the overall optimum. However, the computational time t is likely to increase due to the more complex bottom-up mappings and computations of the optimization driver. In Fig. 6.11 (a), the total computational time t of the approaches (A1), (A2), and (B) can be seen.

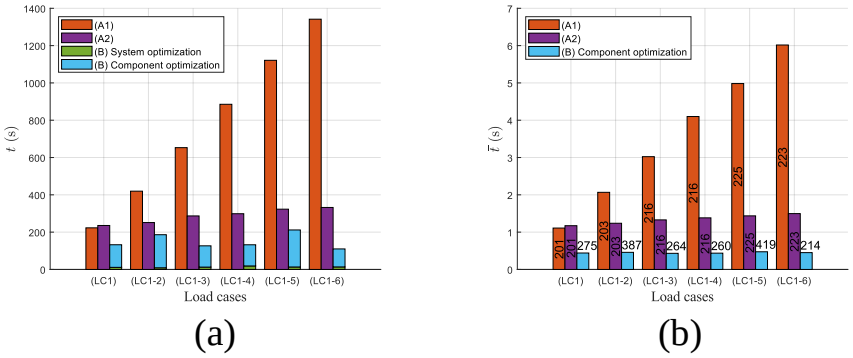


Figure 6.11: Time comparison for (P2.2.2): (a) total computational time t, (b) computational time per iteration $\bar{t}$

For an increasing number of load cases n_p, the computational time $t_{(A1)}$ for approach (A1) now increases almost linearly, which is accordance to the iterative call of the normal bottom-up mapping and the estimation (6.11). In contrast, approach (A2) shows a much flatter slope $t_{(A2)}$ compared to (A1). The expected constant behavior is not achieved, which is due to the higher cost of the optimization driver and the internal coordination of the monolithic architecture. Nevertheless, it is faster than (A1) for n_p>1 load cases. The hierarchical bottom-up mappings of (A2) clearly show the expected advantage here.

Accordingly, the Informed Decomposition (B) also shows lower computational time $t_{(B)}$ than (A1) and is also faster than (A2), showing the lowest computational time for all investigated load cases. The advantage over (A2) can again mainly be explained by the complete independence of the component optimizations and a lower number of design variables for the optimization driver.

In Fig. 6.11 (b), the average computational time $\bar{t}$ and the required iterations n_{iter} are shown. As in Fig. 6.9 (c), (B) shows an almost constant average computational time $\bar{t}_{(B)}$ across all load cases. Thus, the variations in the total computational time $t_{(B)}$ can be explained by different numbers of iterations n_{iter} until convergence. (A2) experienced a small increase per added load case for $\bar{t}$ due to the higher number of constraints of the optimization problem and internal coordination for parallelization. In contrast, (A1) shows a significant increase per new load case c, which is consistent with the linear increase in Fig. 6.11 (a).

It should be noted that the monolithic optimizations (A1) and (A2) for different number of load cases n_p do not converge to the same solution, although (LC1) is the dominant load case for all converged solutions $\boldsymbol{x}^*$. Besides the final mass m, also the number of iterations differ, n_{iter}=[201, 203, 216, 216, 225, 223]. The optimization algorithm had convergence problems for an increasing number of constraints, where the MMA sometimes even failed and more conservative settings of the algorithm were required, e.g., tighter bounds for the moving asymptotes.

The results in terms of the quantities of interest are shown in Table 6.11. Fig. 6.12 shows the final topologies for n_c=4 components and n_p=6 load cases for (A1-2) and (B). Again, all investigated approaches (A1-2) and (B) satisfy the system requirements, with approach (B) also being physically feasible and differing from the benchmark results of (A1-2) by at most 6.44%. Although the different load case configurations affect the monolithic results of (A), (A1) and (A2) always converge to the same solution, once again proving the equivalence of the two approaches.

Table 6.11: Quantities of interest of (P2.2.2) for monolithic optimizations (A1-2) and Informed Decomposition (B)

Number of load cases n_p	Approach	$\sum m_{(i)}$ g	$\frac{(\cdot)-m_A}{m_A}$ %	u mm	Number of load cases n_p	Approach	$\sum m_{(i)}$ g	$\frac{(\cdot)-m_A}{m_A}$ %	u mm
1	(A1)	291.7	-	1.00	2	(A1)	290.4	-	1.00
	(A2)	291.7	-	1.00		(A2)	290.4	-	1.00
	(B)	304.1	4.26	0.99		(B)	309.2	6.44	0.99
3	(A1)	299.5	-	1.00	4	(A1)	299.5	-	1.00
	(A2)	299.5	-	1.00		(A2)	299.5	-	1.00
	(B)	318.7	6.43	1.00		(B)	308.9	3.14	1.00
5	(A1)	299.5	-	1.00	6	(A1)	299.1	-	1.00
	(A2)	299.5	-	1.00		(A2)	299.1	-	1.00
	(B)	311.7	4.06	1.00		(B)	309.1	3.37	1.00

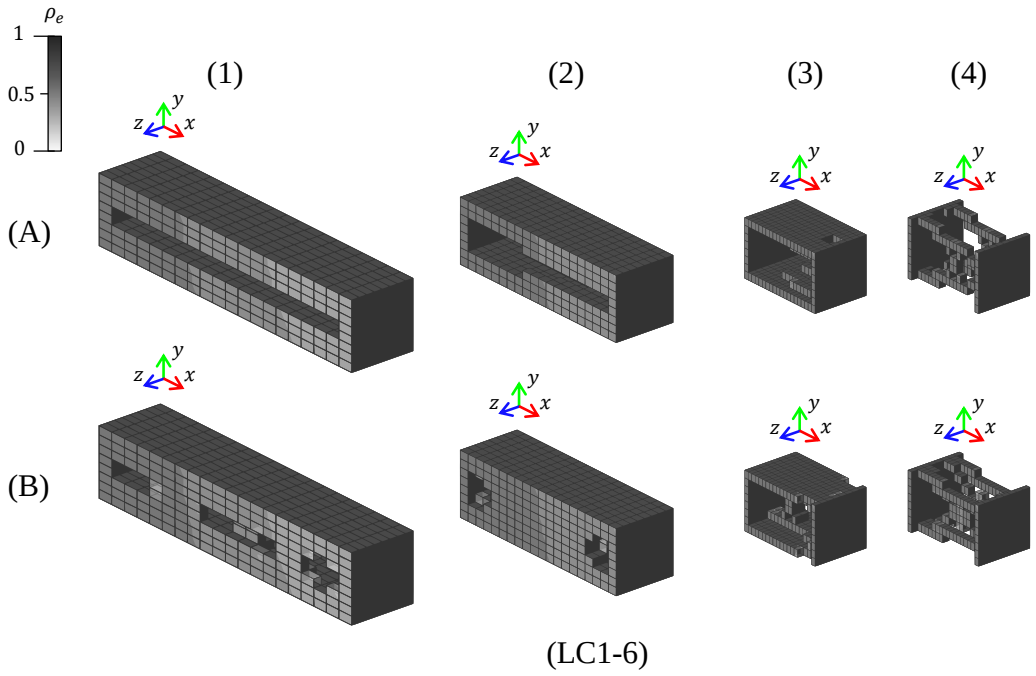


Figure 6.12: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P2.2.2) of the monolithic optimization (A1-2) and Informed Decomposition (B) for n_c=4 components and n_p=6 load cases

Conclusion

The results for the computational time t of the final configuration of (P2.2.1) and (P2.2.2) are shown in Table 6.12. For (P2.2.1), with n_c=4 components for one considered load case, n_p=1, the proposed Informed Decomposition (B) had the lowest computational time $t_{(B)}$. The monolithic approach (A1) was slower by a factor of 1.67 and approach (A2) was even slower by 1.68, showing the slow internal data management of the parallelization. Finally, the ATC (D) showed the highest computational time t and was 22.7 slower than (B).

For (P2.2.2), with n_c=4 components and n_p=6 load cases, approach (B) is faster than (A1) by a factor of 13.9. In addition to the savings of 5.27 expected from the earlier investigation on the different bottom-up mappings of (A1) and (B), see again Table 6.5, the reduced number of design variables for the decoupled component optimizations of (B) reduces the computational time $t_{(B)}$. In contrast, approach (A2) and (B) use the same hierarchical bottom-up mappings that can be parallelized. However, approach (B) is also faster than (A2) by 3.45. The decoupling, without the need to internally coordinate the parallelization, and the low-dimensional optimization problem led to a significant reduction in the computational time t.

Table 6.12: Computational time t comparison between (B), (A1), (A2), and (D) for (P2.2.1) and (P2.2.2)

Design problem (P2.2.1): $n_c = 4$, $n_p = 1$				Design problem (P2.2.2): $n_c = 4$, $n_p = 6$			
(B)	(A1)	(A2)	(D)	(B)	(A1)	(A2)	(D)
\	$\frac{t_{(A1)}}{t_{(B)}} = 1.67$	$\frac{t_{(A2)}}{t_{(B)}} = 1.68$	$\frac{t_{(D)}}{t_{(B)}} = 22.7$	\	$\frac{t_{(A1)}}{t_{(B)}} = 13.9$	$\frac{t_{(A2)}}{t_{(B)}} = 3.45$	-

In conclusion, the approach (A1) with the normal bottom-up mapping scales as expected with the number of components n_c and the number of load cases n_p. The second monolithic approach (A2) also has increasing computational time t for increasing number of components n_c due to internal coordination, but there is only weak coupling with respect to the number of load cases n_p. Approach (D) suffered from significant coordination overhead for all configurations in the first design problem (P2.1.1) and was therefore not investigated further. Only approach (B) showed complete independence from both the number of components n_c and the number of load cases n_p and also did not suffer from coordination overhead. Thus, the decoupled optimization architecture (B) for the design problem (P2.2) was able to reduce the computational time t in comparison to the two given monolithic optimizations (A1-2) as well as ATC (D).

6.4 Design problem (P2.3): Low-cost lightweight robot

6.4.1 Introduction

Design problem (P2.3) deals with the second configuration of Fig. 6.1 (b), a robot architecture with n_{dof}=6 degrees of freedom per interface, where the structural elements of each component i are connected by three-dimensional connectors, see also Fig. 6.13 (a). The robot is designed for a specific trajectory $\boldsymbol{\theta}_{(i)}(t) = [\alpha, 0, 0]^\top$ of a pick-and-place task and a payload of m=1 kg is applied to the end effector, see Fig. 6.13 (b) and Table 6.13. From the given trajectory $\boldsymbol{\theta}_{(i)}(t)$, n_{p}=100 static representative load cases are derived and evaluated with respect to the stiffness requirement $u \leq 1$ mm.

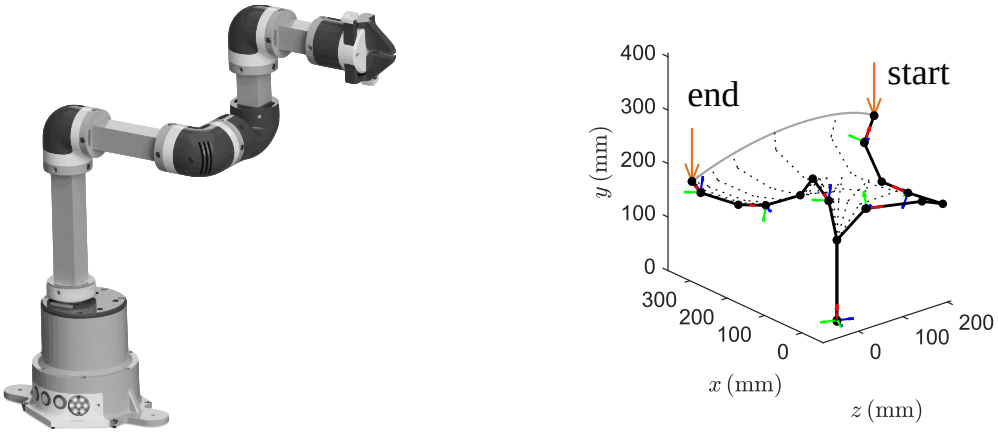


Figure 6.13: (a) Low-cost lightweight robotic architecture in the initial position $\boldsymbol{\theta}_{(i)} = \mathbf{0}$ and (b) the pick-and-place trajectory $\boldsymbol{\theta}_{(i)}(t)$

Table 6.13: Design problem (P2.3): Start and end pose $\boldsymbol{\theta} = [\alpha, \beta, \gamma]^\top$, joint positions $\boldsymbol{p}$, and interface positions $\boldsymbol{a}$ and $\boldsymbol{b}$

	$\boldsymbol{\theta}_{(i)}$ rad	$\boldsymbol{p}_{(0)}$ mm	$\boldsymbol{a}_{(1)}, \boldsymbol{b}_{(1)}$ mm	$\boldsymbol{p}_{(1)}$ mm	$\boldsymbol{a}_{(2)}, \boldsymbol{b}_{(2)}$ mm	$\boldsymbol{p}_{(2)}$ mm	$\boldsymbol{a}_{(3)}, \boldsymbol{b}_{(3)}$ mm	$\boldsymbol{p}_{(3)}$ mm	$\boldsymbol{a}_{(4)}, \boldsymbol{b}_{(4)}$ mm	$\boldsymbol{p}_{(4)}$ mm
Start	$\begin{bmatrix}1.19\\0\\0\end{bmatrix}, \begin{bmatrix}-0.589\\0\\0\end{bmatrix}, \begin{bmatrix}-1.04\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\2\\0\end{bmatrix} \begin{bmatrix}0\\152\\0\end{bmatrix}$	$\begin{bmatrix}-52\\204\\0\end{bmatrix}$	$\begin{bmatrix}-54\\204\\0\end{bmatrix} \begin{bmatrix}-154\\204\\0\end{bmatrix}$	$\begin{bmatrix}-206\\204\\52\end{bmatrix}$	$\begin{bmatrix}-206\\264\\72\end{bmatrix} \begin{bmatrix}-206\\314\\72\end{bmatrix}$	$\begin{bmatrix}-258\\366\\72\end{bmatrix}$	$\begin{bmatrix}-260\\366\\72\end{bmatrix} \begin{bmatrix}-310\\366\\72\end{bmatrix}$	$\begin{bmatrix}-310\\366\\72\end{bmatrix}$
End	$\begin{bmatrix}2.72\\0\\0\end{bmatrix}, \begin{bmatrix}-1.43\\0\\0\end{bmatrix}, \begin{bmatrix}-0.0266\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$									

Note that unlike the earlier investigated design problems, the gravity forces $\boldsymbol{f}_{\text{cog}}$ of the rigid connectors are now also considered, since some of the connectors contain motors and other parts that cannot be neglected. The respective gravity forces at the center of gravity can be recalculated with respect to the interface positions $p_{(i)}$, adding an offset moment for the lever arm distance between the joint position $\boldsymbol{p}$ and the center of gravity $\boldsymbol{x}_{\text{cog}}$, see Fig. 6.14 and Table 6.14. Due to the assumption of rigid joints, this recalculation does not affect the elastic behavior of the multi-component system.

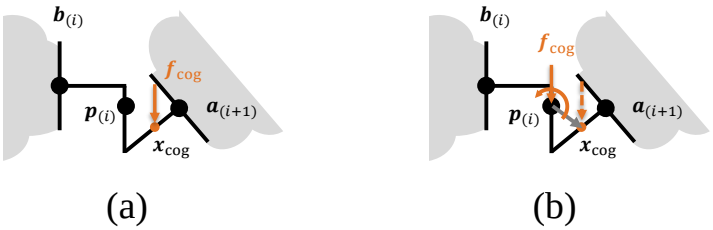


Figure 6.14: (a) Actual acting gravity loads $\boldsymbol{f}_{cog}$ of the rigid connectors and (b) substitute loads acting on the join positions $\boldsymbol{p}_{(i)}$

Table 6.14: Design problem (P2.3): Center of gravity $\boldsymbol{x}_{cog}$ and gravity loads $\boldsymbol{f}_{cog}$ of the connectors for each component i at initial position $\boldsymbol{\theta}_{(i)} = \boldsymbol{0}$, plus payload $\boldsymbol{f}$ at end effector position $\boldsymbol{p}_{(4)}$

Component (1)				Component (2)				Component (3)				Component (4)			
$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{x}_{cog}$	$\boldsymbol{f}_{cog}$	$\boldsymbol{p}_{(4)}$	$\boldsymbol{f}$
mm	N	mm	N	mm	N	mm	N	mm	N	mm	N	mm	N	mm	N
0	0	-18	0	-53	0	-202	0	-206	0	-224	0	-259	0	-310	0
1	0	200	0	204	0	204	0	228	0	362	0	366	0	366	0
0	-0.294	0	-4.90	0	-0.294	18	-4.90	52	-2.35	72	-4.90	72	-0.294	72	-9.81

6.4.2 Offline database

The Informed Decomposition approach (B) can take advantage of the offline database since meta models for l=100 mm already exist. Therefore, component i=2 can reuse the meta models of Section 5.4. For the other three components, two new meta models with input $\boldsymbol{\kappa} \in \mathbb{R}^{1\times 8}$ are needed

$$\hat{p}_{l=50}(\boldsymbol{\kappa}), \quad \hat{p}_{l=150}(\boldsymbol{\kappa}), \tag{6.25}$$

$$\hat{m}_{l=50}(\boldsymbol{\kappa}), \quad \hat{m}_{l=150}(\boldsymbol{\kappa}). \tag{6.26}$$

The parameters for the active-learning sampling can be found in Table 6.15. Once again, the same parameters are used for l=50 mm and l=150 mm as for the already trained meta model of length l=100 mm.

Table 6.15: Overview of the parameters of the active-learning strategy for (i) the classification and (ii) regression sampling phase for (P2.3) for l = 50 mm and l = 150 mm

(i) Classification sampling: $N_J = [2000, 2500, 3000, \ldots, 9000]$ for $J = 1, \ldots, 14$			
Physical seed $\boldsymbol{Y}_\mathrm{p}$	Classification samples $\boldsymbol{Y}_\mathrm{C}$	Temporary samples $\boldsymbol{Y}_\mathrm{t}$	Subset of samples $\boldsymbol{Y}_\mathrm{s}$
$N_\mathrm{p} = 2000$	$N_\mathrm{C} = 9000$	$N_\mathrm{t} = J^4 \frac{\left(10^7 - 10^4\right)}{14^4} + 10^4$	$N_\mathrm{s} = 500$
(ii) Regression sampling: for $A = [C_\mathrm{p} + 1, \ldots, 8000]$			
Feasible classification samples $\boldsymbol{Y}_\mathrm{C} = 1$		Regression samples $\boldsymbol{Y}_\mathrm{R}$	
l = 50 mm	l = 150 mm	l = 50 mm	l = 150 mm
$C_\mathrm{p} = 2668$	$C_\mathrm{p} = 2351$	$N_\mathrm{R} = 8000$	

The overall sample data $[\boldsymbol{Y}, \boldsymbol{Z}]$ for l=50 mm and l=150 mm computed by the active-learning strategy is shown in Fig. 6.15 (a) and (b), respectively. Similar to Section 5.4, the active-learning strategy is not capable of sampling especially the upper bound of the design space for both lengths corresponding to filled design domains Ω.

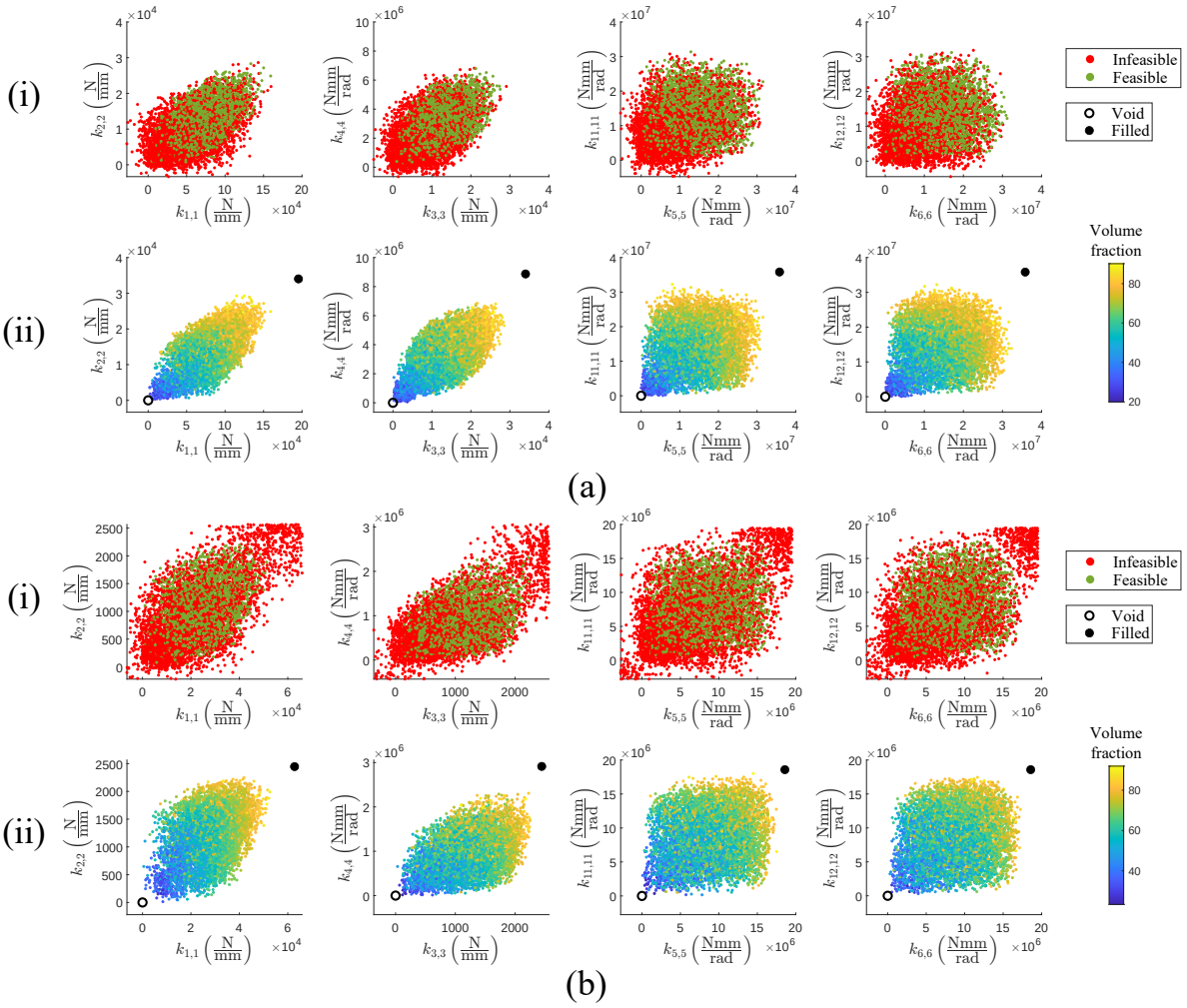


Figure 6.15: Results of the active-learning strategy for (P2.3) for (a) l=50 mm and (b) l=150 mm. The first row are the (i) classification samples for physical feasibility estimation and the second row are the (ii) regression samples for mass estimation

It is also noticeable that during classification phase (i) for l=150 mm, see Fig. 6.15 (b), one of the temporary classifiers failed, resulting in several infeasible sample data points at the upper boundary of the design space. Nevertheless, the sample data is used to train the final meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$. The performance measures for both estimators are listed in Table 6.16. Even though one of the temporary classifiers failed during classification phase (i), the performance measures of the meta models, for l=150 mm, do not appear to be affected.

Table 6.16: Performance measures of $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ for (P2.3)

l=50 mm					l=150 mm				
$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$		$\hat{p}(\boldsymbol{\kappa})$			$\hat{m}(\boldsymbol{\kappa})$	
FPR	TPR	ACC	R^2	MSE (%)	FPR	TPR	ACC	R^2	MSE (%)
0.0393	0.912	0.936	0.973	4.58	0.0473	0.915	0.934	0.935	8.80

6.4.3 System optimization

With the available meta models $\hat{p}(\boldsymbol{\kappa})$ and $\hat{m}(\boldsymbol{\kappa})$ in the offline database, the investigation of (P2.3) can be started. Therefore, approach (A) and (B) are applied to the design problem (P2.3) with n_p=100 static load cases. First, the level (II) κ-representations are computed and can be seen in Table 6.17.

Table 6.17: Component performance (II), $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$, and the quantities on the system level (I), $\boldsymbol{z} = [m, u]$, of design problem (P2.3) for monolithic optimization (A) and the proposed Informed Decomposition (B)

Component-performance level (II): $\boldsymbol{y}_{(i)} = [v_{(i)}, \boldsymbol{\kappa}_{(i)}]$										
			$\boldsymbol{\kappa}_{(i)}$							
Comp. (i)	Approach	$v_{(i)}$	$k_{1,1}$	$k_{2,2}$	$k_{3,3}$	$k_{4,4}$	$k_{5,5}$	$k_{6,6}$	$k_{11,11}$	$k_{12,12}$
		%	$\frac{\mathrm{N}}{\mathrm{mm}}$	$\frac{\mathrm{N}}{\mathrm{mm}}$	$\frac{\mathrm{N}}{\mathrm{mm}}$	$\frac{\mathrm{N\,mm}}{\mathrm{rad}}$	$\frac{\mathrm{N\,mm}}{\mathrm{rad}}$	$\frac{\mathrm{N\,mm}}{\mathrm{rad}}$	$\frac{\mathrm{N\,mm}}{\mathrm{rad}}$	$\frac{\mathrm{N\,mm}}{\mathrm{rad}}$
(1)	(A)	51.92	2.92e4	159	1.32e3	4.10e5	9.26e6	4.93e6	1.09e7	4.63e6
	(B)	60.07	3.23e4	1.47e3	1.17e3	9.51e5	9.80e6	1.24e7	9.44e6	1.19e7
(2)	(A)	50.99	4.32e4	3.68e3	3.68e3	2.99e6	1.46e7	1.46e7	1.39e7	1.39e7
	(B)	50.99	4.17e4	3.24e3	3.24e3	2.67e6	1.33e7	1.25e7	1.20e7	1.22e7
(3)	(A)	34.41	3.73e4	3.96e3	7.51e3	2.64e6	7.72e6	8.53e6	6.88e6	8.02e6
	(B)	38.03	4.94e4	7.24e3	7.65e3	3.15e6	1.09e7	1.02e7	8.27e6	9.74e6
(4)	(A)	16.61	3.12e3	67.0	669	1.30e5	1.61e6	6.05e5	4.00e4	5.74e5
	(B)	28.35	2.22e4	2.94e3	3.82e3	6.92e5	4.70e6	3.37e6	3.77e6	2.80e6

Quantities on the system level (I): $\boldsymbol{z} = [m, u]$				
		$\sum m_{(i)}$	$\frac{(\cdot)-m_A}{m_A}$	$\max(u_c)$
		g	%	mm
	(A)	221.9	-	1.00
	(B)	250.6	12.9	0.99

The Informed Decomposition approach yields results for $\boldsymbol{\kappa}_{(i)}$ that differ significantly for components i=1, 3, and 4, while the second component i=2 has nearly equal values. As in Section 5.4, the system optimization seems to have problems in realizing small stiffness values, e.g., $k_{2,2}$ for $\boldsymbol{\kappa}_{(1)}$, and $k_{2,2}$ and $k_{3,3}$ for $\boldsymbol{\kappa}_{(4)}$. It is possible that the meta models do not have sample data in this region, which prevents the system optimization from computing those values. In summary, the experienced outliers indicate a non-optimal decoupling of the multi-component system by the system optimization of (B).

6.4.4 Component optimization

Next, the optimized $\boldsymbol{\kappa}_{(i)}$ values of level (II) are utilized to derive the final topologies $\boldsymbol{x}_{(i)}$ of level (III) using the decoupled component optimizations (4.37) of (B). The resulting volume fractions $v_{(i)}$ and the quantities $\boldsymbol{z} = [m, u]$ on the system level (I) are listed in Table 6.17. The maximum displacement of the trajectory is denoted as $\max(u_c)$.

It should be noted that the last pose of the given trajectory turned out to be the dominant load case. Therefore, the plausibility is approximately checked by analyzing this load case (LC100), see also Fig. 6.16, and comparing it with the resulting topologies for approach (A) and (B) in Fig. 6.17.

Approach (A), for n_p=100 load cases, shows some specific properties for each component i of the multi-component system. The last component i=4 on which the pay load $\boldsymbol{f}$ acts, has a topology that evolved mainly against a local z-shear force, suggesting that it is not only optimal for the last load case (LC100), but is the dominant load case over the entire discretized trajectory $\boldsymbol{\theta}_{(i)}(t)$. The third

component i=3 exhibits a nearly closed topology, with more material accumulating on the outer edges of the y-axis than on the z-axis. This is again consistent with the orientation of component i=3 for the last pose, which shows mainly loads due to bending about the z-axis, requiring more material at the y-axis boundaries. The additional torsional loads and y-bending moments attempt to close the structure, which also requires material for the z-axis. Component i=2 results in a fully closed tubular profile experiencing the highest torsional load of all components, which can also be concluded from the system pose for the final load case (LC100). Moreover, it exhibits high stiffness to shear and bending loads. Finally, the first component i=1 has a topology that mainly supports bending about the z-axis and normal forces in the local x-axis, yet the open profile lacks stiffness against any torsional load.

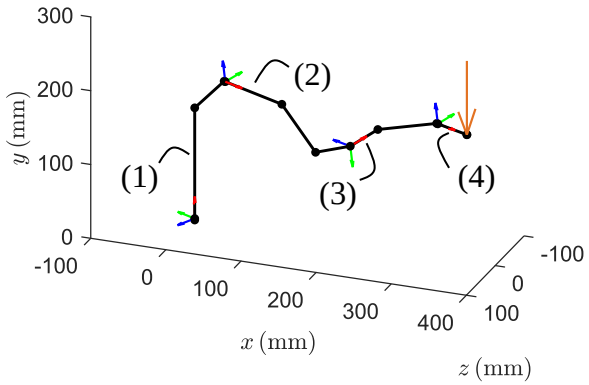


Figure 6.16: Dominant load case (LC100) of the given pick-and-place trajectory

For a static analysis, all topologies are plausible with respect to the last load case (LC100) and hence are also supporting the hypothesis of one dominant load case. However, these results also show a clear drawback of the chosen approach. The static poses of the original dynamic trajectory do not consider dynamic loads. Therefore, especially the first component i=1 shows significant weaknesses in practical applications. The low-cost lightweight robot architecture derived for the given trajectory rotates around the local x-axis of the first component to perform its task. In a dynamic load case, this leads to torsional loads on the first component, for which it provides little stiffness due to its non-closed structure. Nevertheless, for a static consideration, these poses are actually optimal. This is consistent with the idea of a pure top-down design process, where only the explicit requirements for the system are met and thus all implicit requirements are not met.

For the detailed design of approach (B), the components i=1, i=3, and i=4 of the multi-component system significantly deviate from both, the component volume fractions $v_{(i)}$ and their respective topologies $\boldsymbol{x}_{(i)}$ compared to (A). However, the components i=1 and i=3 exhibit the same main characteristics for the experienced loads of the investigated load cases. In contrast, the high deviation of the $\boldsymbol{\kappa}_{(4)}$-vector of component i=4 of the system optimization, also leads to a topology that does not show the expected features against a dominating z-shear load. Additionally, the component mass produced by (B) deviates by 70.7% from (A). In contrast, component i=2 not only agrees for the $\boldsymbol{\kappa}_{(2)}$-values, but also with respect to the topology and the component mass, which turns out to be even identical.

In summary, the total mass deviation is 12.9%, while both approaches (A) and (B) produce feasible designs with respect to the system displacement requirement $u \leq u_{\max}$. Once again, approach (B) had most problems for the component with the lowest volume fraction $v_{(4)}$. The resulting topologies of Fig. 6.17 produced by the proposed Informed Decomposition (B) were subsequently post-processed and mounted into the robot architecture, see Fig. 6.18. The actual low-cost lightweight robot was later built using additive manufacturing in the workshop of the Laboratory of Product Development and Lightweight Design.

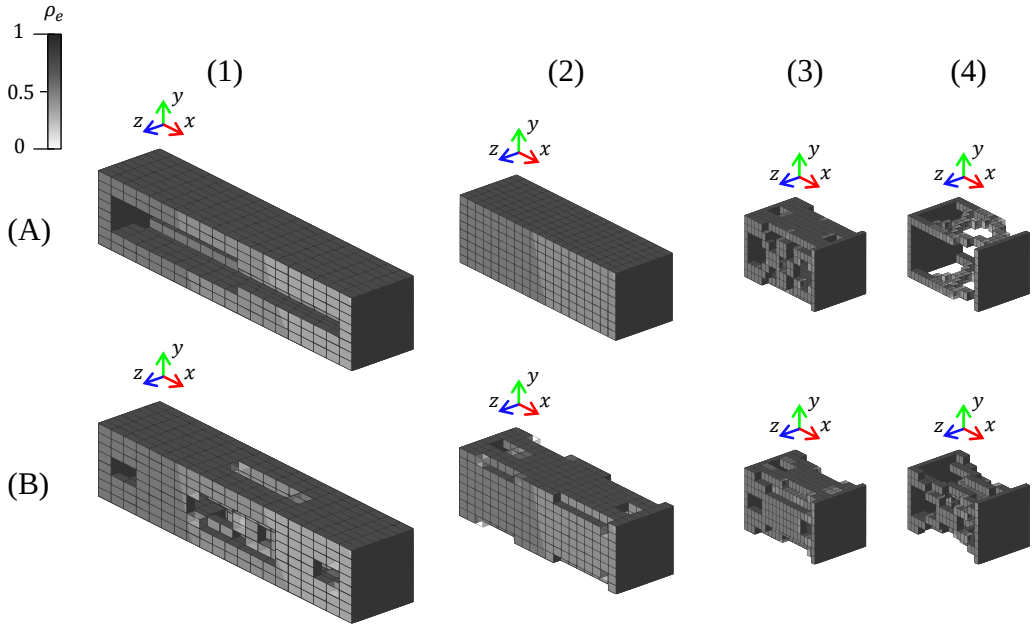


Figure 6.17: Optimized component designs $\boldsymbol{x}_{(i)}$ for (P2.3) of the monolithic optimization (A) and Informed Decomposition (B)

Figure 6.18: Resulting low-cost lightweight robot with topology optimized and post-processed structural elements for n_p=100 load cases

7 Discussion

The main goal of this thesis was to develop a new hierarchical and decoupled distributed optimization architecture for the lightweight design of mechanical multi-component systems referred to as Informed Decomposition (B). In the following, a general discussion on the results is given in order to show both advantages and disadvantages of the developed approach.

Physical feasibility and optimality. The validity of the developed approach was investigated by analyzing the obtained results with respect to physical feasibility and optimality of the system mass. Therefore, the results were compared to the ones of a classical monolithic optimization problem. This was done by slowly increasing the number of interface degrees of freedom from two, to three, and finally to six. It could be observed that as the number was increased, the result quality deteriorated also, see Table 5.12. However, all results were physically feasible and also feasible with respect to the system stiffness requirement. The result quality and hence the validity of the developed approach depends mainly on the utilized meta models, which are responsible for distributing the stiffness properties to the components in a physically feasible and mass-optimal way. They are trained on an increasing number of input dimensions, which makes it more and more difficult to sample the entire space thoroughly. Hence, the sample data provided by the active-learning strategy deteriorates and therefore also the generalization of the meta models. However, the results were also compared to an uninformed decomposition, where the components were decoupled based on rough estimations and then just individually optimized. Here, one component was even physically infeasible and also an infeasible system design with respect to the stiffness requirement occurred. The deviations of the system mass, i.e., the non-optimality, for the uninformed decomposition was significantly higher than for the proposed approach. Thus, the key component of using meta models proved to be crucial in the context of decoupled optimization architectures. The uninformed approach ran the risk of both proposing physically infeasible designs and failing to provide mass-optimal solutions.

Offline database. Generating sample data for the meta models of the offline database is the critical step within the proposed approach. Each sample point requires computationally expensive component optimization, resulting in high sampling cost. The implemented active-learning undersampling strategy helps to enable an efficient sampling procedure while providing a well-balanced dataset. During the sampling procedure, a trade-off must be made between high accuracy (low ϵ in component optimization) and low computational cost (high ϵ). In general, it is not guaranteed that the entire design space is actually covered during the sampling procedure. The results showed that especially stiffness regions for low and high volume fractions often have predictions of lower quality or are falsely estimated as physically infeasible. Furthermore, the lower and upper bound of the design space were never fully captured. This worsens for components with a higher number of degrees of freedom per interface and hence higher dimensions of the input design space due to the curse of dimensionality. One main problem is the selection mechanism of new data points of the active-learning strategy, where only the distance to the separating hyperplane is considered. Especially distances for higher dimensions tend to be equal making a differentiation of designs difficult. Also, no mechanism for evenly distributed points is included, hence some regions are preferred by the algorithm just by the nature of the physical feasible design space.

In the field of optimization architectures, there are approaches that make this expensive sampling and training process a part of the optimization itself. However, in other applications as in the context of multi-scale optimization, meta models that are pre-trained and part of an offline database are also used. Following the idea of an offline database, the Informed Decomposition approach utilizes pre-trained meta models, which are supposed to be used for a variety of different design problems. Similitude theory allows the reuse of meta models for same detailed designs on different geometrical design

domains. However, so far this is only possible with constant scaling in all coordinate directions. If an arbitrary change is desired, the meta models must be explicitly extended by that information. This is an important step towards building an offline database that can be used for arbitrary mechanical design problems, making the need for a new expensive training process very unlikely. So far, only the application to one additional input and two degrees of freedom per interface has been shown, limiting the current approach.

Product development processes. One main motivation for distributed optimization architectures is to mimic the structure of classic product development processes, where different departments can design their subsystems independently (Martins & Ning, 2022). These classical product development processes usually prohibit the use of monolithic architectures, which require a central decision unit and a company-wide software architecture. Therefore, decomposition of distributed optimization architectures should support design optimization to be more easily deployed in industry. Nevertheless, distributed architectures have rarely been used. One reason is that although classical distributed architectures decompose a single optimization problem into a set of smaller optimization subproblems, they still need to be coordinated by a system level problem and also need to communicate with each other. In an industrial context, this coordination would also require an optimization architecture for all departments of a company, which is difficult to realize. Therefore, decoupling as implemented in the proposed Informed Decomposition, without the need for coordination should simplify an application in industry, since after the system optimization all problems can be solved completely independently.

Computational time. The second motivation for distributed architectures is to reduce computational time by decomposing large-size monolithic optimization problems. However, most distributed architectures cannot reduce computational time in general compared to monolithic optimization, but only for specific design problems (Martins & Ning, 2022). In this work, different load cases in linear statics for a mechanical multi-component system were investigated. For the given design problems, a detailed investigation of the computational cost was performed by comparing the resulting computational time with two monolithic approaches as well as analytical target cascading as a hierarchical distributed optimization architecture. Among the architectures studied, Informed Decomposition showed the lowest computational time required to solve the given design problems. The computational time of Informed Decomposition proved to be completely independent of the number of components and the number of load cases, and also did not suffer from coordination overhead. The computational time advantage can be explained mainly by two reasons: (1) the hierarchically decomposed bottom-up mappings used, (2) the decoupled optimization architecture that allows parallelization of the independent component optimizations without an overarching software architecture. First, (1) the hierarchical bottom-up mappings allow the computational cost of the bottom-up mappings to be approximately independent of both the number of components and the number of load cases. Second, (2) the decoupling of Informed Decomposition enables the decomposition of optimization subproblems without coordination or an overarching software architecture. In comparison, monolithic approaches can also use decomposed bottom-up mappings to reduce computational time, but without decomposition, the size of the optimization problem remains the same and an overarching software architecture is required. Classical distributed architectures, on the other hand, also decompose the problem and thus reduce the size. However, they still require an overarching architecture for coordination and therefore run the risk that the coordination time exceeds the time of the original optimization problem, i.e., coordination overhead. Moreover, in this work, internal coordination for parallelization for monolithic and distributed architectures also affected the computational time. This is avoided by the decoupling of the proposed approach, which further simplifies and also speeds up the optimization. It should be noted, however, that the expensive training of meta models has been neglected in this computational time investigation. This assumption is only valid if an offline database has been built beforehand and the approach can be in fact applied to a wide range of design problems.

Multidisciplinary design optimization. So far, only serial mechanical multi-component systems with two system quantities, i.e., system mass and displacement, within linear statics were investigated. In general, an extension to other quantities of interests or even other physical domains should be possible. As long as quantities can be formulated that are component inherent, the theoretical framework remains the same. More difficult are local quantities, such as stresses, which can only be computed at the detail level. The system optimization cannot explicitly account for such local quantities, so there is a risk of again computing a physically infeasible design. Apart from the extension to different quantities of interests and/or disciplines, distributed architecture are classically applied to multiple different disciplines. This so-called aspect partitioning, meaning the decomposition is carried out by discipline boundaries, instead of the physical components is the main application of multidisciplinary design optimization. The application of the Informed Decomposition to original multidisciplinary design optimization problems is also possible, if each discipline can be represented by one or multiple meta models. Even though the investigated design problem is a complicating constraints problem, also optimization problems with other characteristics should be possible.

Mesh resolution. The finite element models used in this work only have coarse meshes. Nevertheless, the aforementioned computational time advantage of a decoupled optimization architecture remains also for finer meshes, but the sampling method in particular would become more expensive. Theoretically, the decoupled component optimizations involving the detailed designs could also be performed with finer meshes than those used for the meta models. The system optimization using these meta models would then decouple the system in a non-optimal but possibly still sufficient way. The finer meshes of the decoupled component optimizations can then only lead to equivalent or better designs than the reference design of the coarse mesh.

Post-processing. Finally, the results of topology optimization often require a post-processing step. On the one hand, gray scaling may occur, where the engineer has to make sense of the results after topology optimization. On the other hand, the geometrical design domain is usually approximated by finite elements and often needs to be converted to a CAD format for subsequent design steps. In this work, only manual post-processing based on image smoothing was performed, see Fig. 6.18. However, small changes in the final topology might cause an invalid component behavior violating system requirements. Therefore, a requirement-based post-processing is necessary to reliably transfer the results into an actual feasible multi-component system.

8 Conclusion

8.1 Summary

In this thesis, the lightweight design of mechanical multi-component systems in linear statics was addressed. In order to design the lightest possible system, structural optimization techniques are often used, such as topology optimization. However, the monolithic design of large systems with many interacting components is a difficult task. First, the design process itself requires decomposition to enable separate and independent development. Second, the size of the system may make overall monolithic optimization prohibitively expensive, requiring decomposition to reduce the size and the computational time. Therefore, distributed optimization architectures have been developed in which the monolithic optimization problem is decomposed into a set of smaller optimization subproblems containing subsets of the objectives, design variables, and constraints. In practice, however, distributed optimization architectures have not been able to adequately address these two existing problems and are therefore rarely used. A major reason for this is that the decomposed optimization problems of most distributed architectures are not fully separable and require a coordination strategy to maintain consistency between the subproblems. This complicates the application of such architectures in an industrial context that requires a department-wide software architecture. In addition, using such a coordination strategy risks that the actual coordination cost exceeds the cost of the original optimization problem, which is called coordination overhead and prevents these architectures from being faster.

For this thesis, the lightweight design task involves a serial mechanical multi-component system, where the first component is clamped, and a static payload is applied on the last component. The system must sustain a vertical load with a maximum translational end effector displacement for minimum mass. The systems under investigation can be hierarchically divided into three levels. The main drivers of complexity are the number of interface degrees of freedom, the number of components, and the number of load cases investigated. To solve this lightweight design task, the Informed Decomposition approach was proposed in Chapter 4. This hierarchical design optimization approach consists of

(a) a decoupled optimization architecture consisting of a surrogate-based system optimization that decouples the problem and subsequent independent and parallel component optimizations, and
(b) an offline database consisting of feasibility estimators and mass estimators.

The decoupled optimization architecture (a) first performs a system optimization that minimizes the system mass while satisfying the system displacement requirement. Moreover, the system optimization decouples the given design problem with respect to the physical components, i.e., an object-based partitioning. Unlike classical decomposition schemes, decoupling decomposes the monolithic optimization into independent optimization subproblems that can be solved without coordination. The performance of each mechanical component of the system is defined by interface stiffness matrices and their respective component masses. Since the detailed structure of each component is not known a priori, the physical feasibility of each interface stiffness matrix is also considered. A component is considered physically feasible if a given component performance can be subsequently realized at the detail level. This is achieved by providing the system optimization with meta models that evaluate both the physical feasibility and the mass of the given component stiffness matrices. The surrogate-based system optimization thus assigns the required stiffness properties for each component in a mass-optimal and physically feasible manner without knowing their detailed descriptions.

In a second step, the optimized stiffness properties are used to formulate constraints for the subsequent component optimizations. The component optimizations minimize the mass for the given stiffness properties and can be performed completely independently, eliminating the need for coordination between components or the system level. Decoupling also allows parallel component optimizations

that do not require an overarching software architecture. Instead of the actual interface stiffness matrix, the elastic properties of the stiffness matrices are condensed without loss of information into a low-dimensional κ-representation. This representation is an important part of the proposed approach as it provides a link between the different hierarchical levels and also serves as a communication variable between the system optimization and the component optimizations.

Since training meta models for mass and feasibility estimation is computationally expensive, the Informed Decomposition approach is also based on the idea of an offline database (b). This means that training of meta models is not part of the optimization architecture (a), but that already trained meta models are available that can be used for the majority of given design problems. Thus, training of meta models does not need to be performed for each new design problem, but only when no suitable models are available in the database. The appropriate meta models also have higher applicability due to similitude theory for similar geometrical design domains. If arbitrary geometrical changes of the design domain are needed, the meta models can be explicitly extended to geometrical information. In general, not all combinations of stiffness entries are physically feasible. Therefore, each randomly selected dataset is assumed to contain many more infeasible than feasible data points. So-called imbalanced training data degrades the performance of meta models and makes the sampling process less efficient. Therefore, an active-learning undersampling strategy was developed to enable an efficient sampling process while maintaining a well-balanced dataset. The sampling strategy consists of two phases: (i) classification sampling and (ii) regression sampling. Classification sampling (i) approximates the hyperplane between feasible and infeasible designs by selecting sample points according to intermediate classifiers. In regression sampling (ii), the trained classifier evaluates regression samples so that only feasible samples are optimized and assumed infeasible sample points are ignored a priori.

After having established the decoupled optimization architecture, the following Chapter 5 examined the validity of the approach. In order to verify the validity, the physical feasibility and optimality with respect to mass were investigated by analyzing a linear mechanical two-component system. During the investigation, the complexity was slowly increased with respect to the number of load cases as well as the interface degrees of freedom. The results were compared to the ones of a classical monolithic optimization. Overall, the results were all physically feasible and also feasible with respect to the system stiffness requirement. However, as the number of degrees of freedom increases, the result quality deteriorates slightly from 0.40% to 3.91% to a maximum of 8.18% mass deviation for the last investigation.

Afterwards, important aspects for practical application are presented in Chapter 6. Therefore, a linear four-component system with different lengths was investigated. For the first design problem, the system was studied with only one set of meta models for all four components with different lengths. This section therefore showed how the scope of meta models can be extended to support the construction of an offline database of multiple meta models that can be used for any mechanical design problem. The second design problem included a detailed investigation on the computational time of the proposed Informed Decomposition approach and compared the results to two monolithic approaches as well as one other distributed optimization architecture, namely analytical target cascading. It could be shown that the computational time of optimization problems can be reduced significantly by utilizing a decoupled architecture. A hierarchical and decomposed bottom-up mapping can reduce the computational cost related to the number of components and load cases, while an appropriate optimization architecture also affects the convergence behavior and internal coordination processes. Finally, the last design problem deals with a low-cost lightweight robot that was designed for a given specific trajectory of a pick-and-place task. Hence, 100 static representative load cases of the given dynamic trajectory were derived and evaluated with respect to the system stiffness requirement. The results produced for this robot application deviate no more than 12.9% in mass from the monolithic benchmark result and were all physically feasible. The resulting topologies were subsequently post-processed and mounted into a robot architecture that was later built using additive manufacturing in the workshop of the Laboratory of Product Development and Lightweight Design.

8.2 Outlook

Some promising potentials for further improvements are outlined in the following. First, as already discussed in Section 7, the active-learning undersampling strategy used could be modified to explicitly account for the distribution of sample points within the design space, as is the case with latin hypercube sampling, for example. This could improve the quality of point selection and thus reduce the observed degradation of results for higher input dimensions of the meta models, especially for the full six degrees of freedom per interface.

Second, the system optimization is based on a gradient-free and global search particle swarm optimization. Although this algorithm has proven successful for a variety of design problems, it relies on some heuristic parameters and requires multiple runs due to its statistical nature without the ability to accurately replicate previous results. A gradient-based algorithm could support the repeatability and reproducibility of the presented approach, but with the higher risk of convergence to a local optimum.

Third, the component performance was realized by topology optimization for rectangular cross sections. Since classical SIMP-based topology optimization works on pre-defined geometrical design domains discretized by finite elements, an extension to other cross sections, such as circular profiles, could be implemented, too. It would also be interesting to use different approaches to structural optimization, such as size or shape optimization, and compare the results with topology optimization. This would only require a change in the component optimization without affecting the general framework of Informed Decomposition. The component optimization itself uses a target stiffness as a constraint that can be processed in different ways. Up to now, a two-sided inequality constraint has been used to realize approximately the required component stiffness with a small deviation of ϵ. However, this imposes an unnecessarily stringent requirement on the component stiffness, since it also excludes designs that are too stiff. A one-sided formulation could solve this problem by using more general stiffness measures, such as the eigenvalues of the respective stiffness matrices.

Despite further development of the approach itself, an extension to design problems with other system characteristics should also be performed to prove the general validity of the developed optimization architecture. An extension to parallel mechanical multi-component systems should be possible without changes within the method. Also, components with a higher number of interfaces could be used, but a new κ-representation has to be created here.

For industrial relevance, further application examples should be considered. So far, only the last design problem of a low-cost lightweight robot represents a practical application. Since multi-component systems can be found in many industrial fields, such as suspensions in automotive engineering, truss structures in civil engineering, landing gears in aerospace engineering, there is a wide range of possible application scenarios. Different application scenarios may also require different quantities of interest. Up to now, only displacement quantities have been studied, so an investigation of other quantities of interest and/or disciplines is important. The moment of inertia has been investigated by several researchers using topology optimization since voxel-based approaches allow for efficient inertia computation. This would pave the way towards dynamic investigations. Moreover, the general structure of heat conduction problems is similar to that of compliance-based topology optimization in linear statics, indicating a reasonable extension to another discipline.

Finally, an investigation of classical multidisciplinary design optimization problems involving different disciplines could be conducted. Here, a comparison between the established distributed architectures and the proposed Informed Decomposition for specific benchmark problems would further validate the approach.

References

Alexandrov, N. M., & Hussaini, M. Y. (1997). *Multidisciplinary design optimization: state-of-the-art*. Philadelphia: Society for Industrial and Applied Mathematics SIAM.

Andersen, L., & Nielsen, S. R. (2008). *Elastic beams in three dimensions: textbook* (23rd ed.). Aalborg: Aalborg University.

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine Learning*, *50*(1/2), 5–43. doi: 10.1023/A:1020281327116

Bathe, K.-J. (2014). *Finite element procedures* (Second edition ed.). Englewood Cliffs, N.J: Prentice-Hall.

Bendsøe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, *71*(2), 197–224. doi: 10.1016/0045-7825(88)90086-2

Bendsøe, M. P., & Sigmund, O. (2004). *Topology optimization: Theory, methods, and applications* (Second Edition, Corrected Printing ed.). Berlin and Heidelberg: Springer. doi: 10.1007/978-3-662-05086-6

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009) ed.). New York, NY: Springer New York.

Boolchandani, D., Ahmed, A., & Sahula, V. (2011). Efficient kernel functions for support vector machine regression model for analog circuits' performance evaluation. *Analog Integrated Circuits and Signal Processing*, *66*(1), 117–128. doi: 10.1007/s10470-010-9476-6

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory - colt '92*. New York, New York, USA: ACM Press. doi: 10.1145/130385.130401

Braun, R. (1996). *Collaborative optimization: an architecture for large-scale distributed design: Ph.d. thesis*.

Braun, R., Gage, P., Kroo, I. M., & Sobieski, I. (1996). Implementation and performance issues in collaborative optimization. In *6th symposium on multidisciplinary analysis and optimization.* Reston, Virigina: American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1996-4017

Braun, R., & Kroo, I. M. (1997). Development and application of the collaborative optimization architecture in a multidisciplinary design environment. *Multidisciplinary design optimization: state of the art*.

Casaburo, A., Petrone, G., Franco, F., & de Rosa, S. (2019). A review of similitude methods for structural engineering. *Applied Mechanics Reviews*, *71*(3). doi: 10.1115/1.4043787

Conejo, A. J., Nogales, F. J., & Prieto, F. J. (2002). A decomposition procedure based on approximate newton directions. *Mathematical Programming*, *93*(3), 495–515. doi: 10.1007/s10107-002-0304-3

Corinna Cortes, & Vladimir Vapnik. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. doi: 10.1007/BF00994018

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (Fourth edition ed.). Cambridge, Massachusett: The MIT Press.

Coutinho, C. P., Baptista, A. J., & Dias Rodrigues, J. (2016). Reduced scale models based

on similitude theory: A review up to 2015. *Engineering Structures*, *119*, 81–94. doi: 10.1016/j.engstruct.2016.04.016

Deaton, J. D., & Grandhi, R. V. (2014). A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, *49*(1), 1–38. doi: 10.1007/s00158-013-0956-z

Ding, M., & Vemur, R. I. (2005). An active learning scheme using support vector machines for analog circuit feasibility classification. *Proceedings of the 18th International Conference on VLSI Design*, 528–534. doi: 10.1109/ICVD.2005.47

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). Piscataway, NJ: IEEE Service Center. doi: 10.1109/MHS.1995.494215

Eckert, C., & Clarkson, J. (2005). The reality of design. In J. Clarkson & C. Eckert (Eds.), *Design process improvement* (pp. 1–29). London: Springer London. doi: 10.1007/978-1-84628-061-0_1

Ertekin, S., Huang, J., Bottou, L., & Giles, L. (2007). Learning on the border. In M. J. Silva et al. (Eds.), *Proceedings of the 2007 acm conference on information and knowledge management* (p. 127). New York, NY: ACM. doi: 10.1145/1321440.1321461

Ferrer, A., Oliver, J., Cante, J. C., & Lloberas-Valls, O. (2016). Vademecum-based approach to multi-scale topological material design. *Advanced Modeling and Simulation in Engineering Sciences*, *3*(1), 23. doi: 10.1186/s40323-016-0078-4

Forsberg, K., & Mooz, H. (1991). The relationship of system engineering to the project cycle. *INCOSE International Symposium*, *1*(1), 57–65. doi: 10.1002/j.2334-5837.1991.tb01484.x

Gilbert, J. R., Moler, C., & Schreiber, R. (1992). Sparse matrices in matlab: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, *13*(1), 333–356. doi: 10.1137/0613024

Guyan, R. J. (1965). Reduction of stiffness and mass matrices. *AIAA Journal*, *3*(2), 380. doi: 10.2514/3.2874

Haftka, R. T., & Watson, L. T. (2005). Multidisciplinary design optimization with quasiseparable subsystems. *Optimization and Engineering*, *6*(1), 9–20. doi: 10.1023/B:OPTE.0000048534.58121.93

Han, J., & Papalambros, P. Y. (2010). A note on the convergence of analytical target cascading with infinite norms. *Journal of Mechanical Design*, *132*(3). doi: 10.1115/1.4001001

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Second edition ed.). New York: Springer. doi: 10.1007/978-0-387-84858-7

Haykin, S. S. (2009). *Neural networks and learning machines* (3. ed. ed.). New York: Pearson.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. doi: 10.1109/TKDE.2008.239

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. Mahwah, N.J: L. Erlbaum Associates. doi: 10.4324/9781410612403

Heirman, G. H. K., & Desmet, W. (2010). Interface reduction of flexible bodies for efficient modeling of body flexibility in multibody dynamics. *Multibody System Dynamics*, *24*(2), 219–234. doi: 10.1007/s11044-010-9198-7

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. doi: 10.1016/0893-6080(89)90020-8

Hou, L., & Jiao, R. J. (2020). Data-informed inverse design by product usage information: a review, framework and outlook. *Journal of Intelligent Manufacturing*, *31*(3), 529–552. doi: 10.1007/s10845-019-01463-2

Huang, S., & Schimmels, J. M. (1998). Achieving an arbitrary spatial stiffness with springs connected in parallel. *Journal of Mechanical Design*, *120*(4), 520–526. doi: 10.1115/1.2829309

Huang, S., & Schimmels, J. M. (2000). The eigenscrew decomposition of spatial stiffness matrices. *IEEE Transactions on Robotics and Automation*, *16*(2), 146–156. doi: 10.1109/70.843170

Hughes, T. J. R. (2000). *The finite element method: Linear static and dynamic finite element analysis*. Mineola, NY: Dover Publications.

Irons, B. (1963). Eigenvalue economisers in vibration problems. *The Journal of the Royal Aeronautical Society*, *67*(632), 526–528. doi: 10.1017/S0001924000062618

Irons, B. (1965). Structural eigenvalue problems - elimination of unwanted variables. *AIAA Journal*, *3*(5), 961–962. doi: 10.2514/3.3027

James, G. (2013). *An introduction to statistical learning: With applications in r* (Vol. 103). New York, NY: Springer. doi: 10.1007/978-1-4614-7138-7

Jeong, S.-H., Choi, D.-H., & Jeong, M. (2012). Feasibility classification of new design points using support vector machine trained by reduced dataset. *International Journal of Precision Engineering and Manufacturing*, *13*(5), 739–746. doi: 10.1007/s12541-012-0096-1

Jung, J., Yoon, J. I., Park, S.-J., Kang, J.-Y., Kim, G. L., Song, Y. H., ... Kim, H. S. (2019). Modelling feasibility constraints for materials design: Application to inverse crystallographic texture problem. *Computational Materials Science*, *156*, 361–367. doi: 10.1016/j.commatsci.2018.10.017

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings / 1995 ieee international conference on neural networks* (pp. 1942–1948). Piscataway, NJ: IEEE. doi: 10.1109/ICNN.1995.488968

Kerscher, T. (2021). *Machine learning estimators for structural optimization* (Master's thesis). Technical University of Munich.

Kim, B. J., Yun, D. K., Lee, S. H., & Jang, G.-W. (2016). Topology optimization of industrial robots for system-level stiffness maximization by using part-level metamodels. *Structural and Multidisciplinary Optimization*, *54*(4), 1061–1071. doi: 10.1007/s00158-016-1446-x

Kim, H. M., Michelena, N. F., Papalambros, P. Y., & Jiang, T. (2003). Target cascading in optimal system design. *Journal of Mechanical Design*, *125*(3), 474–480. doi: 10.1115/1.1582501

Kim, J.-G., & Lee, P.-S. (2014). An accurate error estimator for guyan reduction. *Computer Methods in Applied Mechanics and Engineering*, *278*, 1–19. doi: 10.1016/j.cma.2014.05.002

Kollmann, H. T., Abueidda, D. W., Koric, S., Guleryuz, E., & Sobh, N. A. (2020). Deep learning for topology optimization of 2d metamaterials. *Materials & Design*, *196*(4), 109098. doi: 10.1016/j.matdes.2020.109098

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232. doi: 10.1007/s13748-016-0094-0

Kremer, J., Steenstrup Pedersen, K., & Igel, C. (2014). Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(4), 313–326. doi:

10.1002/widm.1132

Krischer, L., Sureshbabu, A. V., & Zimmermann, M. (2020). Modular topology optimization of a humanoid arm: 2020 3rd international conference on control and robots (iccr). doi: 10.1109/ICCR51572.2020.9344316

Krischer, L., Sureshbabu, A. V., & Zimmermann, M. (2022). Active-learning combined with topology optimization for top-down design of multi-component systems. *Proceedings of the Design Society*, *2*, 1629–1638. doi: 10.1017/pds.2022.165

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

Kroo, I. M. (1997). Mdo for large-scale design. *Multidisciplinary design optimization: state of the art*, 22–44.

Kulick, J., Lang, T., Toussaint, M., & Lopes, M. (2013). Active learning for teaching a robot grounded relational symbols: International joint conference on artificial intelligence.

Lei, X., Liu, C., Du, Z., Zhang, W., & Guo, X. (2019). Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics*, *86*(1), 1. doi: 10.1115/1.4041319

Li, Z. (2022). *Topology optimization using machine learning models* (Semester's thesis). Technical University of Munich.

Liu, B., Haftka, R. T., & Watson, L. T. (2004). Global-local structural optimization using response surfaces of local optimization margins. *Structural and Multidisciplinary Optimization*, *27*(5). doi: 10.1007/s00158-004-0393-0

Liu, G.-R., & Quek, S. S. (2013). *The finite element method: A practical course* (Second ed.). Oxford, UK: Butterworth-Heinemann.

Martins, J. R. R. A., & Lambe, A. B. (2013). Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, *51*(9), 2049–2075. doi: 10.2514/1.J051895

Martins, J. R. R. A., & Ning, A. (2022). *Engineering design optimization*. Cambridge and New York, NY: Cambridge University Press.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133. doi: 10.1007/BF02478259

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239. doi: 10.2307/1268522

Meyer, C. D. (2008). *Matrix analysis and applied linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics.

Michell, A. (1904). The limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *8*(47), 589–597. doi: 10.1080/14786440409463229

Milton, G., & Cherkaev, A. V. (1995). Which elasticity tensors are realizable? *Journal of Engineering Materials and Technology*, *117*(4), 483–493. doi: 10.1115/1.2804743

Milton, G., Harutyunyan, D., & Briane, M. (2017). Towards a complete characterization of the effective elasticity tensors of mixtures of an elastic phase and an almost rigid phase. *Mathematics and Mechanics of Complex Systems*, *5*(1), 95–113. doi: 10.2140/memocs.2017.5.95

Minsky, M., & Papert, S. A. (1969). *Perceptrons: An introduction to computational geometry* (2. print. with corr ed.). Cambridge/Mass.: The MIT Press.

Mukherjee, S., Lu, D., Raghavan, B., Breitkopf, P., Dutta, S., Xiao, M., & Zhang, W. (2021). Accelerating large-scale topology optimization: State-of-the-art and challenges. *Archives of Computational Methods in Engineering*, *28*(7), 4549–4571. doi: 10.1007/s11831-021-09544-3

Oh, S., Jung, Y., Kim, S., Lee, I., & Kang, N. (2019). Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, *141*(11), 436. doi: 10.1115/1.4044229

Pan, V. (1987). Complexity of parallel matrix computations. *Theoretical Computer Science*, *54*(1), 65–85. doi: 10.1016/0304-3975(87)90019-3

Papadrakakis, M., Lagaros, N. D., & Tsompanakis, Y. (1998). Structural optimization using evolution strategies and neural networks. *Computer Methods in Applied Mechanics and Engineering*, *156*(1-4), 309–333. doi: 10.1016/s0045-7825(97)00215-6

Papalambros, P. Y., & Wilde, D. J. (2018). *Principles of optimal design*. Cambridge University Press. doi: 10.1017/9781316451038

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, *1*(1), 33–57. doi: 10.1007/s11721-007-0002-0

Qiu, C., Han, Y., Shanmugam, L., Zhao, Y., Dong, S., Du, S., & Yang, J. (2021). A deep learning-based composite design strategy for efficient selection of material and layup sequences from a given database. *Composites Science and Technology*, 109154. doi: 10.1016/j.compscitech.2021.109154

Ramu, P., Thananjayan, P., Acar, E., Bayrak, G., Park, J. W., & Lee, I. (2022). A survey of machine learning techniques in structural and multidisciplinary optimization. *Structural and Multidisciplinary Optimization*, *65*(9). doi: 10.1007/s00158-022-03369-9

Regenwetter, L., & Ahmed, F. (2022). Design target achievement index: A differentiable metric to enhance deep generative models in multi-objective inverse design. *arXiv preprint arXiv:2205.03005.*. Retrieved from `https://arxiv.org/pdf/2205.03005`

Rinner, T. (2022). *Comparison of a machine learning based distributed topology optimization scheme to monolithic optimization* (Master's thesis). Technical University of Munich.

Robinson, A. L., Taub, A. I., & Keoleian, G. A. (2019). Fuel efficiency drives the auto industry to reduce vehicle weight. *MRS Bulletin*, *44*(12), 920–923. doi: 10.1557/mrs.2019.298

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. doi: 10.1037/h0042519

Rozvany, G. I. N. (2009). A critical review of established methods of structural topology optimization. *Structural and Multidisciplinary Optimization*, *37*(3), 217–237. doi: 10.1007/s00158-007-0217-0

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *The 1998 ieee international conference on evolutionary computation proceedings* (pp. 69–73). Piscataway, NJ: Inst. of Electrical and Electronics Engineers. doi: 10.1109/ICEC.1998.699146

Sigmund, O. (2011). On the usefulness of non-gradient approaches in topology optimization. *Structural and Multidisciplinary Optimization*, *43*(5), 589–596. doi: 10.1007/s00158-011-0638-7

Sigmund, O., & Maute, K. (2013). Topology optimization approaches. *Structural and Multidisciplinary*

Optimization, *48*(6), 1031–1055. doi: 10.1007/s00158-013-0978-6

Sigmund, O., & Petersson, J. (1998). Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural and Multidisciplinary Optimization*, *16*(1), 68–75. doi: 10.1007/BF01214002

Sobieszczanski-Sobieski, J., Agte, J., & Sandusky, R. R. (2000). Bilevel integrated system synthesis. *AIAA Journal*, *38*(1), 164–172. doi: 10.2514/2.937

Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., & Sandusky, R. R. (2003). Bilevel integrated system synthesis for concurrent and distributed processing. *AIAA Journal*, *41*(10), 1996–2003. doi: 10.2514/2.1889

Sobieszczanski-Sobieski, J., & Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural and Multidisciplinary Optimization*, *14*(1), 1–23. doi: 10.1007/BF01197554

Sosnovik, I., & Oseledets, I. (2019). Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, *34*(4), 215–223. doi: 10.1515/rnam-2019-0018

Svanberg, K. (1987). The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, *24*(2), 359–373. doi: 10.1002/nme.1620240207

Svanberg, K. (2007). *Mma and gcmma-two methods for nonlinear optimization*.

Thomas, C. T. (2022). *Decomposition and structural optimization of systems using machine learning models for varying geometrical dimensions* (Semester's thesis). Technical University of Munich.

Tosserams, S., Etman, L. F. P., Papalambros, P. Y., & Rooda, J. E. (2006). An augmented lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and Multidisciplinary Optimization*, *31*(3), 176–189. doi: 10.1007/s00158-005-0579-0

Tosserams, S., Etman, L. F. P., & Rooda, J. E. (2009). A classification of methods for distributed system optimization based on formulation structure. *Structural and Multidisciplinary Optimization*, *39*(5), 503–517. doi: 10.1007/s00158-008-0347-z

Ulu, E., Zhang, R., & Kara, L. B. (2015). A data-driven investigation and estimation of optimal topologies under variable loading configurations. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, *4*(2), 61–72. doi: 10.1080/21681163.2015.1030775

Vapnik, V., & Chervonenkis, A. (1974). *Theory of pattern recognition*. Moscow: Nauka.

Vapnik, V., & Kotz, S. (2006). *Estimation of dependences based on empirical data* (2nd ed. ed.). New York, NY: Springer New York.

Viana, F. A. C., Simpson, T. W., Balabanov, V., & Toropov, V. (2014). Special section on multidisciplinary design optimization: Metamodeling in multidisciplinary design optimization: How far have we really come? *AIAA Journal*, *52*(4), 670–690. doi: 10.2514/1.J052375

Wang, L., Chan, Y.-C., Ahmed, F., Liu, Z., Zhu, P., & Chen, W. (2020). Deep generative modeling for mechanistic-based learning and design of metamaterial systems. *Computer Methods in Applied Mechanics and Engineering*, *372*, 113377. doi: 10.1016/j.cma.2020.113377

Wang, L., Tao, S., Zhu, P., & Chen, W. (2021). Data-driven topology optimization with multiclass microstructures using latent variable gaussian process. *Journal of Mechanical Design*, *143*(3), 296. doi: 10.1115/1.4048628

Wang, M., Song, Y., Lian, B., Wang, P., Chen, K., & Sun, T. (2022). Dimensional parameters and structural topology integrated design method of a planar 5r parallel machining robot. *Mechanism and Machine Theory*, *175*, 104964. doi: 10.1016/j.mechmachtheory.2022.104964

Wang, X., Zhang, D., Zhao, C., Zhang, P., Zhang, Y., & Cai, Y. (2019). Optimal design of lightweight serial robots by integrating topology optimization and parametric system optimization. *Mechanism and Machine Theory*, *132*, 48–65. doi: 10.1016/j.mechmachtheory.2018.10.015

White, D. A., Arrighi, W. J., Kudo, J., & Watts, S. E. (2019). Multiscale topology optimization using neural network surrogate models. *Computer Methods in Applied Mechanics and Engineering*, *346*, 1118–1135. doi: 10.1016/j.cma.2018.09.007

Woldseth, R. V., Aage, N., Bærentzen, J. A., & Sigmund, O. (2022). On the use of artificial neural networks in topology optimisation. *Structural and Multidisciplinary Optimization*, *65*(10). doi: 10.1007/s00158-022-03347-1

Wu, J., Sigmund, O., & Groen, J. P. (2021). Topology optimization of multi-scale structures: a review. *Structural and Multidisciplinary Optimization*, *63*(3), 1455–1480. doi: 10.1007/s00158-021-02881-8

Wu, Z., Xia, L., Wang, S., & Shi, T. (2019). Topology optimization of hierarchical lattice structures with substructuring. *Computer Methods in Applied Mechanics and Engineering*, *345*(2), 602–617. doi: 10.1016/j.cma.2018.11.003

Xia, L., & Breitkopf, P. (2015). Multiscale structural topology optimization with an approximate constitutive model for local material microstructure. *Computer Methods in Applied Mechanics and Engineering*, *286*, 147–167. doi: 10.1016/j.cma.2014.12.018

Yilin, G., Fuh Ying Hsi, J., & Wen Feng, L. (2021). Multiscale topology optimisation with nonparametric microstructures using three-dimensional convolutional neural network (3d-cnn) models. *Virtual and Physical Prototyping*, *16*(3), 306–317. doi: 10.1080/17452759.2021.1913783

Yu, Y., Hur, T., Jung, J., & Jang, I. G. (2019). Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, *59*(3), 787–799. doi: 10.1007/s00158-018-2101-5

Zienkiewicz, O. C., Taylor, R. L., & Zhu, J. Z. (2005). *The finite element method: Its basis and fundamentals* (6th ed. ed.). Amsterdam and Boston: Elsevier Butterworth-Heinemann.

Zimmermann, M., Königs, S., Niemeyer, C., Fender, J., Zeherbauer, C., Vitale, R., & Wahle, M. (2017). On the design of large systems subject to uncertainty. *Journal of Engineering Design*, *28*(4), 233–254. doi: 10.1080/09544828.2017.1303664

Zimmermann, M., & von Hoessle, J. E. (2013). Computing solution spaces for robust design. *International Journal for Numerical Methods in Engineering*, *94*(3), 290–307. doi: 10.1002/nme.4450

List of Figures

List of Tables

A Appendices

A.1 Finite element mesh numbering conventions

The utilized three-dimensional brick elements $\boldsymbol{K}_e$ with the respective local node numbering convention can be seen in Fig. A.1

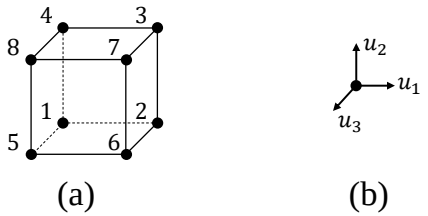


Figure A.1: Three-dimensional brick elements with the local (a) node numbering convention and (b) n_{dof}=3 translational degrees of freedom per node

and the global numbering conventions of the finite element model with respect to the finite elements and nodes of the mesh are shown in Fig. A.2.

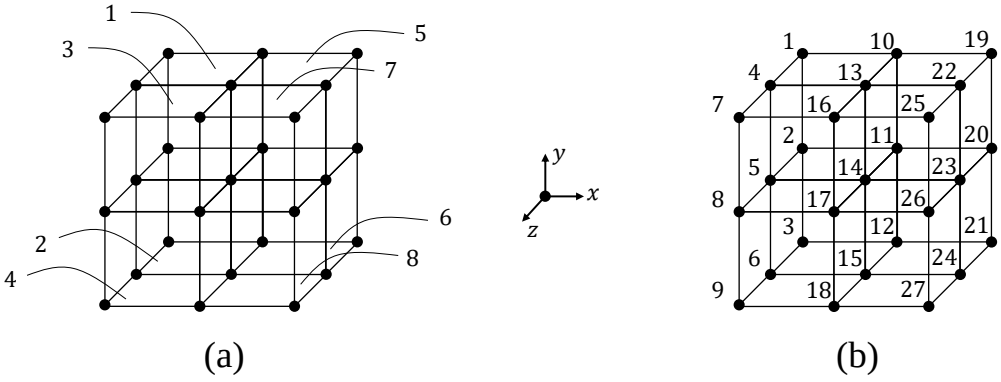


Figure A.2: Exemplary finite element mesh for $n_{ele} = [2, 2, 2]^\top$ with the respective global (a) element numbering and (b) node numbering convention

A.2 RBE2 formulations

A.2.1 Interface condensation matrix

In order to connect the left and right side, $\boldsymbol{A}$ and $\boldsymbol{B}$, to the interface degrees of freedom of $\boldsymbol{a}$ and $\boldsymbol{b}$, the following geometrical constraint matrix $\boldsymbol{C}$ is established

$$\boldsymbol{C}=\begin{bmatrix}
1 & 0 & 0 & 0 & \Delta z & -\Delta y & -1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -\Delta z & 0 & \Delta x & 0 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta y & -\Delta x & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & \Delta z & -\Delta y & 0 & 0 & 0 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -\Delta z & 0 & \Delta x & 0 & 0 & 0 & 0 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta y & -\Delta x & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & \vdots & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta z & -\Delta y \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\Delta z & 0 & \Delta x \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta y & -\Delta x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta z & -\Delta y \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & -\Delta z & 0 & \Delta x \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & \Delta y & -\Delta x & 0
\end{bmatrix}, \quad \text{(A.1)}$$

where the first half is related to the left side $\boldsymbol{A}$ and the second half to the right side $\boldsymbol{B}$. The constraint matrix $\boldsymbol{C}$ then needs to be ordered with respect to the master and slave degrees of freedom, m and s, respectively, to compute the interface condensation matrix $\boldsymbol{T}_{\mathrm{r}}$

$$\boldsymbol{C}=\begin{bmatrix}\boldsymbol{C}_m & \boldsymbol{C}_s\end{bmatrix}, \quad \boldsymbol{T}_{\mathrm{r}}=\begin{bmatrix}\boldsymbol{I} \\ -\boldsymbol{C}_s^{-1}\boldsymbol{C}_m\end{bmatrix}.$$

A.2.2 Joint condensation matrix

To assemble the multi-component system, the left and right side, $\boldsymbol{a}_{(i)}$ and $\boldsymbol{b}_{(i)}$, are connected rigidly to its respective joint positions $\boldsymbol{p}_{(i-1)}$ and $\boldsymbol{p}_{(i)}$. In contrast to the original formulation of Section 2.1.3, here no explicit new degrees of freedom are introduced for the constraint matrix $\boldsymbol{C}$, but the interface points are just recomputed with respect to the joint positions

$$\boldsymbol{C}=\begin{bmatrix}
1 & 0 & 0 & 0 & \Delta z & -\Delta y & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -\Delta z & 0 & \Delta x & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta y & -\Delta x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta z & -\Delta y \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\Delta z & 0 & \Delta x \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta y & -\Delta x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad \text{(A.2)}$$

and the reduced joint condensation matrix $\boldsymbol{T}_{\mathrm{p},(i-1,i)}$ is just

$$\boldsymbol{T}_{\mathrm{p},(i-1,i)}=\boldsymbol{C}. \quad \text{(A.3)}$$

A.3 Explicit linear mapping

For a mechanical body with n_{int}=2 interfaces and n_{dof}=6 degrees of freedom per interface, n_ϕ=6 rigid body modes exist

$$\boldsymbol{\phi}_{(1)} = \begin{bmatrix} u & 0 & 0 & 0 & 0 & 0 & u & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top, \tag{A.4}$$

$$\boldsymbol{\phi}_{(2)} = \begin{bmatrix} 0 & u & 0 & 0 & 0 & 0 & 0 & u & 0 & 0 & 0 & 0 \end{bmatrix}^\top, \tag{A.5}$$

$$\boldsymbol{\phi}_{(3)} = \begin{bmatrix} 0 & 0 & u & 0 & 0 & 0 & 0 & 0 & u & 0 & 0 & 0 \end{bmatrix}^\top, \tag{A.6}$$

$$\boldsymbol{\phi}_{(4)} = \begin{bmatrix} 0 & 0 & 0 & \varphi & 0 & 0 & 0 & 0 & 0 & \varphi & 0 & 0 \end{bmatrix}^\top, \tag{A.7}$$

$$\boldsymbol{\phi}_{(5)} = \begin{bmatrix} 0 & 0 & u & 0 & \varphi & 0 & 0 & 0 & -u & 0 & \varphi & 0 \end{bmatrix}^\top, \quad \varphi = 2u/l, \tag{A.8}$$

$$\boldsymbol{\phi}_{(6)} = \begin{bmatrix} 0 & -u & 0 & 0 & 0 & \varphi & 0 & u & 0 & 0 & 0 & \varphi \end{bmatrix}^\top, \quad \varphi = 2u/l, \tag{A.9}$$

see also Fig 4.3.

The independent entries of the given interface stiffness matrix $\boldsymbol{K}$ for $x-y$ and $x-z$ symmetry planes

$$\boldsymbol{K} = \begin{bmatrix} k_{1,1} & 0 & 0 & 0 & 0 & 0 & k_{1,7} & 0 & 0 & 0 & 0 & 0 \\ & k_{2,2} & 0 & 0 & 0 & k_{2,6} & 0 & k_{2,8} & 0 & 0 & 0 & k_{2,12} \\ & & k_{3,3} & 0 & k_{3,5} & 0 & 0 & 0 & k_{3,9} & 0 & k_{3,11} & 0 \\ & & & k_{4,4} & 0 & 0 & 0 & 0 & 0 & k_{4,10} & 0 & 0 \\ & & & & k_{5,5} & 0 & 0 & 0 & k_{5,9} & 0 & k_{5,11} & 0 \\ & & & & & k_{6,6} & 0 & k_{6,8} & 0 & 0 & 0 & k_{6,12} \\ & & & & & & k_{7,7} & 0 & 0 & 0 & 0 & 0 \\ & & & \text{sym} & & & & k_{8,8} & 0 & 0 & 0 & k_{8,12} \\ & & & & & & & & k_{9,9} & 0 & k_{9,11} & 0 \\ & & & & & & & & & k_{10,10} & 0 & 0 \\ & & & & & & & & & & k_{11,11} & 0 \\ & & & & & & & & & & & k_{12,12} \end{bmatrix},$$

can be reduced utilizing equation (4.6)

$$\boldsymbol{K}\boldsymbol{\phi}_r = \mathbf{0}.$$

If it is solved with respect to

$$\boldsymbol{\kappa}_{\text{sym}} = \begin{bmatrix} k_{1,1}, k_{2,2}, k_{3,3}, k_{4,4}, k_{5,5}, k_{6,6}, k_{11,11}, k_{12,12} \end{bmatrix} \in \mathbb{R}^{1\times 8},$$

the following relations hold for the explicit linear mapping $\boldsymbol{\Phi}_{\text{sym}}(\boldsymbol{\kappa})=\boldsymbol{K}$

$$\begin{array}{llll} k_{1,1} = \kappa_1, & k_{1,7} = -\kappa_1, & & \\ k_{2,2} = \kappa_2, & k_{2,6} = \frac{\kappa_2 l^2 + \kappa_6 - \kappa_8}{2l}, & k_{2,8} = -\kappa_2, & k_{2,12} = \frac{\kappa_2 l^2 - \kappa_6 + \kappa_8}{2l}, \\ k_{3,3} = \kappa_3, & k_{3,5} = -\frac{\kappa_3 l^2 + \kappa_5 - \kappa_7}{2l}, & k_{3,9} = -\kappa_3, & k_{3,11} = -\frac{\kappa_3 l^2 - \kappa_5 + \kappa_7}{2l}, \\ k_{4,4} = \kappa_4, & k_{4,10} = -\kappa_4, & & \\ k_{5,5} = \kappa_5, & k_{5,9} = \frac{\kappa_3 l^2 + \kappa_5 - \kappa_7}{2l}, & k_{5,11} = \frac{\kappa_3 l^2 - \kappa_5 - \kappa_7}{2}, & \\ k_{6,6} = \kappa_6, & k_{6,8} = -\frac{\kappa_2 l^2 + \kappa_6 - \kappa_8}{2l} & k_{6,12} = \frac{\kappa_2 l^2 - \kappa_6 - \kappa_8}{2} & \\ k_{7,7} = \kappa_1, & & & \\ k_{8,8} = \kappa_2, & k_{8,12} = -\frac{\kappa_2 l^2 - \kappa_6 + \kappa_8}{2l}, & & \\ k_{9,9} = \kappa_3, & k_{9,11} = \frac{\kappa_3 l^2 - \kappa_5 + \kappa_7}{2l}, & & \\ k_{10,10} = \kappa_4, & & & \\ k_{11,11} = \kappa_7, & & & \\ k_{12,12} = \kappa_8. & & & \end{array} \tag{A.10}$$

B Nomenclature

$\boldsymbol{a}$	$\in \mathbb{R}^{3\times 1}$	Left interface position
$\boldsymbol{b}$	$\in \mathbb{R}^{3\times 1}$	Right interface position
$\boldsymbol{d}$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times 1}$	Displacement vector
$\boldsymbol{d}_{\text{s}}$	$\in \mathbb{R}^{n_{\text{s}}\times 1}$	System displacement vector
$\boldsymbol{f}$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times 1}$	Load vector
$\boldsymbol{f}_{\text{s}}$	$\in \mathbb{R}^{n_{\text{s}}\times 1}$	System load vector
h	$\in \mathbb{R}^{1\times 1}$	Height of design domain
$\boldsymbol{K}$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times(n_{\text{int}}n_{\text{dof}})}$	Interface stiffness matrix
$\boldsymbol{K}_{\text{d}}$	$\in \mathbb{R}^{n_{\text{d}}\times n_{\text{d}}}$	Detailed stiffness matrix
$\boldsymbol{K}_e$	$\in \mathbb{R}^{24\times 24}$	Element stiffness matrix of a translational brick element
$\boldsymbol{K}_{\text{p}}$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times(n_{\text{int}}n_{\text{dof}})}$	Joint stiffness matrix
$\boldsymbol{K}_{\text{s}}$	$\in \mathbb{R}^{n_{\text{s}}\times n_{\text{s}}}$	System stiffness matrix
l	$\in \mathbb{R}^{1\times 1}$	Length of design domain
m	$\in \mathbb{R}^{1\times 1}$	System mass
$\hat{m}$	$\in \mathbb{R}^{1\times 1}$	Mass estimator
$\boldsymbol{p}$	$\in \mathbb{R}^{3\times 1}$	Joint position
$\hat{p}$	$\in \mathbb{R}^{1\times 1}$	Feasibility estimator
$\boldsymbol{T}_{\text{g}}$	$\in \mathbb{R}^{n_{\text{d}}\times(n_A+n_B+n_{\text{int}}n_{\text{dof}})}$	Guyan condensation matrix
$\boldsymbol{T}_{\text{p}}$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times(n_{\text{int}}n_{\text{dof}})}$	Joint condensation matrix
$\boldsymbol{T}_{\text{r}}$	$\in \mathbb{R}^{(n_A+n_B+n_{\text{int}}n_{\text{dof}})\times(n_{\text{int}}n_{\text{dof}})}$	Interface condensation matrix
u	$\in \mathbb{R}^{1\times 1}$	Euclidean norm of translational end effector displacement vector
$\boldsymbol{u}$	$\in \mathbb{R}^{3\times 1}$	Translational displacement vector
$\boldsymbol{u}_{\text{ee}}$	$\in \mathbb{R}^{3\times 1}$	Translational end effector displacement vector
v	$\in \mathbb{R}^{1\times 1}$	Volume fraction
w	$\in \mathbb{R}^{1\times 1}$	Width of design domain
$\boldsymbol{x}$	$\in \mathbb{R}^{1\times n_{\text{x}}}$	Quantities on component-detail level (III)
$\boldsymbol{Y}$	$\in \mathbb{R}^{N\times n_{\text{y}}}$	Input sample data
$\boldsymbol{y}$	$\in \mathbb{R}^{1\times n_{\text{y}}}$	Quantities on component-performance level (II)
$\boldsymbol{Z}$	$\in \mathbb{R}^{N\times n_{\text{z}}}$	Output sample data
$\boldsymbol{z}$	$\in \mathbb{R}^{1\times n_{\text{z}}}$	Quantities on system level (I)
$\boldsymbol{\alpha}$	$\in \mathbb{R}^{3\times 1}$	Scaling factor
$\boldsymbol{\theta}$	$\in \mathbb{R}^{3\times 1}$	Joint rotations
$\boldsymbol{\kappa}$	$\in \mathbb{R}^{1\times n_{\text{k}}}$	Low-dimensional representation of the interface stiffness matrix
$\boldsymbol{\lambda}$	$\in \mathbb{R}^{1\times n_{\text{k}}}$	Scaling parameter
ρ	$\in \mathbb{R}^{1\times 1}$	Material density
ρ_e	$\in \mathbb{R}^{1\times 1}$	Element density of SIMP-based topology optimization
$\boldsymbol{\varphi}$	$\in \mathbb{R}^{3\times 1}$	Rotational displacement vector
$\boldsymbol{\phi}_r$	$\in \mathbb{R}^{(n_{\text{int}}n_{\text{dof}})\times 1}$	Rigid body mode
$\boldsymbol{\Phi}(\boldsymbol{\kappa})$		Linear mapping function
Ω		Design domain
$(\cdot)_{\text{lb/ub}}$		Lower/upper bound of variables
$(\cdot)^t$		Target values
$(\cdot)^*$		Functions or variables at their optimal value
$\hat{(\cdot)}$		Meta models of a given function
$\tilde{(\cdot)}$		Functions or variables that are copies
$(\cdot)_{(i)}$		Functions or variables that apply only to one component
$(\cdot)_{(0)}$		Functions or variables shared by more than one component

C Glossary

Architecture	Combination of coordination strategy and one or more problem formulations
Bottom-up mapping	Mapping of lower-level design variables onto higher-level outputs, often also called analysis
Component	A component consists of a structural element and rigid connectors on both sides
Coordination overhead	Coordination cost exceeds the computational cost of the original monolithic optimization problem
Coordination strategy	A coordination strategy determines the information exchange within a given optimization problem
Decomposition	Partitioning of a design problem into smaller subproblems
Decoupling	Decoupling is a decomposition without the need for coordination
Design domain	Geometrical space of a structural element which is designed by detailed design variables
Design space	Input space defined by a set of design variables that may be restricted by bounds or constraints
Distributed architecture	Architectures that decompose an optimization problem into smaller optimization subproblems
End effector	A device at the end of a multi-component system used to perform functions, mainly used in robotics
Interface	An idealized point of connection and interaction between two or more elements
Interface stiffness matrix	A stiffness matrix that determines the elastic behavior of a structural element with respect to interfaces
Load case	Combination of system pose and acting system load
Monolithic architecture	Architectures that form and solve a single optimization problem, may include several bottom-up mappings
Multi-component system	A collection of mechanical components that interact with each other and perform specified functions
Offline database	Database with pre-trained meta models that can then be reused for different design problems
Physically feasible	A proposed top-level design that can be realized on a detail level afterwards
Rigid body element	An idealized element that does not deform under external loads or displacements
Scaling	Changing the geometrical dimensions of a structural element by a constant factor
System pose	Prescribed set of joint rotations that define the position and orientation of a multi-component system