

## Inhaltsverzeichnis

<b>1 Einleitung.....</b>	<b>3</b>
1.1 Ausgangssituation .....	3
1.2 Problemstellung und Motivation .....	4
1.3 Zielsetzung der Arbeit .....	5
1.4 Struktur der Arbeit.....	6
<b>2 Stand der Wissenschaft und Technik .....</b>	<b>7</b>
2.1 Grundlagen der Strukturmechanik.....	7
2.1.1 Finite-Elemente-Analyse (FEM) .....	7
2.1.2 Guyan-Reduktion.....	10
2.1.3 Starrkörper-Elemente.....	11
2.1.4 $\kappa$ -Representation.....	12
2.2 Machine learning .....	13
2.2.1 Support-Vector-Machine .....	14
2.2.2 Neural Nets .....	18
2.3 Strukturoptimierung.....	23
<b>3 Allgemeine Methodik.....</b>	<b>25</b>
3.1 Problemstellung .....	25
3.2 Sample-Erzeugung.....	25
3.3 Systemoptimierung .....	30
3.4 Komponentenoptimierung .....	31
<b>4 1-D Balken .....</b>	<b>32</b>
4.1 Problemdefinition .....	32
4.2 Ergebnisse .....	34
4.2.1 Parameterstudie.....	34
4.2.2 Endergebnisse .....	41
<b>5 2D Topologie.....</b>	<b>45</b>

5.1 Problemdefinition.....	45
5.2 Ergebnisse .....	48
5.2.1 Parameterstudie.....	48
5.2.2 Endergebnisse .....	52
<b>6 3D Topologie .....</b>	<b>57</b>
6.1 Problemdefinition.....	57
6.2 Ergebnisse .....	58
6.2.1 Parameterstudie.....	58
6.2.2 Endergebnisse .....	59
<b>7 Diskussion .....</b>	<b>66</b>
<b>8 Zusammenfassung und Ausblick.....</b>	<b>67</b>
<b>9 Literaturverzeichnis.....</b>	<b>68</b>
<b>Anhang .....</b>	<b>A-1</b>
A1 2D Spannungselementsteifigkeitsmatrix .....	A-2
A2 $\kappa$ -Representation mit 6 Freiheitsgrad.....	A-3

# 1 Einleitung

Topologieoptimierung ist ein populärer Zweig der Mechanik, bei dem es darum geht, eine möglichst leichte Struktur zu erreichen und gleichzeitig die mechanischen Anforderungen zu erfüllen.

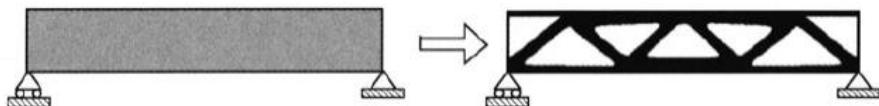


Abb. 1-1: Ein Basisstrukturprofil und die entsprechende topologisch optimierte Struktur (Bendsøe & Sigmund, 2004, S.2)

Es gibt heutzutage eine Vielzahl verschiedener Methoden zur Topologieoptimierung, von denen die meisten auf den Arbeiten von Bendsøe und Kikuchi (1988) beruhen und die in zahlreichen Fällen Anwendung finden. Das aus der Optimierung resultierende Design wird nicht direkt in der Produktion eingesetzt, sondern kann als Referenz für das zu produzierende Design verwendet werden. Es bildet einen Ausgangspunkt für die weitere Gestaltungsarbeit der Ingenieure (Bendsøe & Sigmund, 2004, S. 24-25).

## 1.1 Ausgangssituation

In der Regel wird bei der Topologieoptimierung, auch als *monolithische Optimierung* bekannt, versucht, ein System ganzheitlich zu lösen. Allerdings ist dies in manchen Situationen nicht möglich, sodass das Problem zerlegt werden muss. Zu solchen Situationen gehört beispielsweise eine große Anzahl von Komponenten oder Materialien in einem System. Darüber hinaus werden die Komponenten eines Systems in der Regel von separaten Entwicklungsabteilungen entworfen (Martins und Lambe 2013; Zimmermann et al. 2017). Wenn die verschiedenen Komponenten entkoppelt und anschließend separat optimiert werden können, kommt dies der unabhängigen Arbeit der einzelnen Abteilungen zugute.

Ein alternativer Ansatz zur topologischen Optimierung ist die verteilte Optimierung. Hierbei wird das gesamte Problem in Teilprobleme unterteilt, von denen jedes über ein eigenes Objekt sowie eigene Designvariablen und Einschränkungen verfügt. Hierzu wird von Krischer et al. (2020) eine Methode zur effektiven Optimierung eines Roboterarms mit zwei vollständig entkoppelten Komponenten vorgeschlagen. Deren Idee besteht darin, die globale Beschränkung der Armauslenkung zu verwenden, die jeder Komponente einen bestimmten relativen Anteil an der Steifigkeit des Gesamtsystems zuweist, und daraufhin jede Komponente separat zu optimieren. Der Ansatz der verteilten Optimierung zerlegt große Probleme in kleinere Teilprobleme, erfordert aber typischerweise einen intensiven Koordinationsaufwand (Krischer & Zimmermann, 2021).

Um dieses Koordinationsproblem zu vermeiden, wurde von Zimmermann und von Hoessle (2013) ein neues quantitatives Top-Down-Design zur Erreichung einer vollständigen

Entkopplung eingeführt, das in Zimmermann et al. (2017) in den Designprozess eingebettet ist. Darin werden ausgehend von den Systemobjekten sogenannte *Designräume* berechnet, was sich um eine Reihe von Komponenteneigenschaften handelt, die in dieser Arbeit durch den  $\kappa$ -Vektor nach Vereinfachung der Komponentensteifigkeitsmatrix repräsentiert werden. Die näherungsweise quadratischen Designräume stellen zulässige Intervalle für Komponenteneigenschaften dar und können zur Formulierung vollständig entkoppelter Komponentenanforderungen verwendet werden (Krischer & Zimmermann, 2021).

Dieses Top-Down-Design birgt jedoch die Gefahr in sich, die optimale, aber zuvor unbekannte Lösung im Voraus auszuschließen, die auch als *Loss of Solution Space* (Zimmermann and von Hoessle, 2013) bekannt ist. Um das zu vermeiden, wird ein weiterer Begriff, *Metamodelle*, eingeführt. Metamodelle haben eine lange Geschichte im Systemdesign, da sie die Rechenkosten umfangreicher Computermodelle reduzieren können (Viana et al., 2014). Sie werden im Top-Down-Design eingeführt, um *A-priori-Informationen* der Komponentenebene einzubeziehen, um Loss of Solution Space zu verhindern (Krischer & Zimmermann, 2021). Da einige Informationen über das System vor der Optimierung desselben erhalten werden, wird das Optimierungsverfahren als *Informed Decomposition* bezeichnet (Krischer & Zimmermann, 2021). Dies ist die Strukturoptimierungsmethode, die in dieser Arbeit verwendet wird.

## 1.2 Problemstellung und Motivation

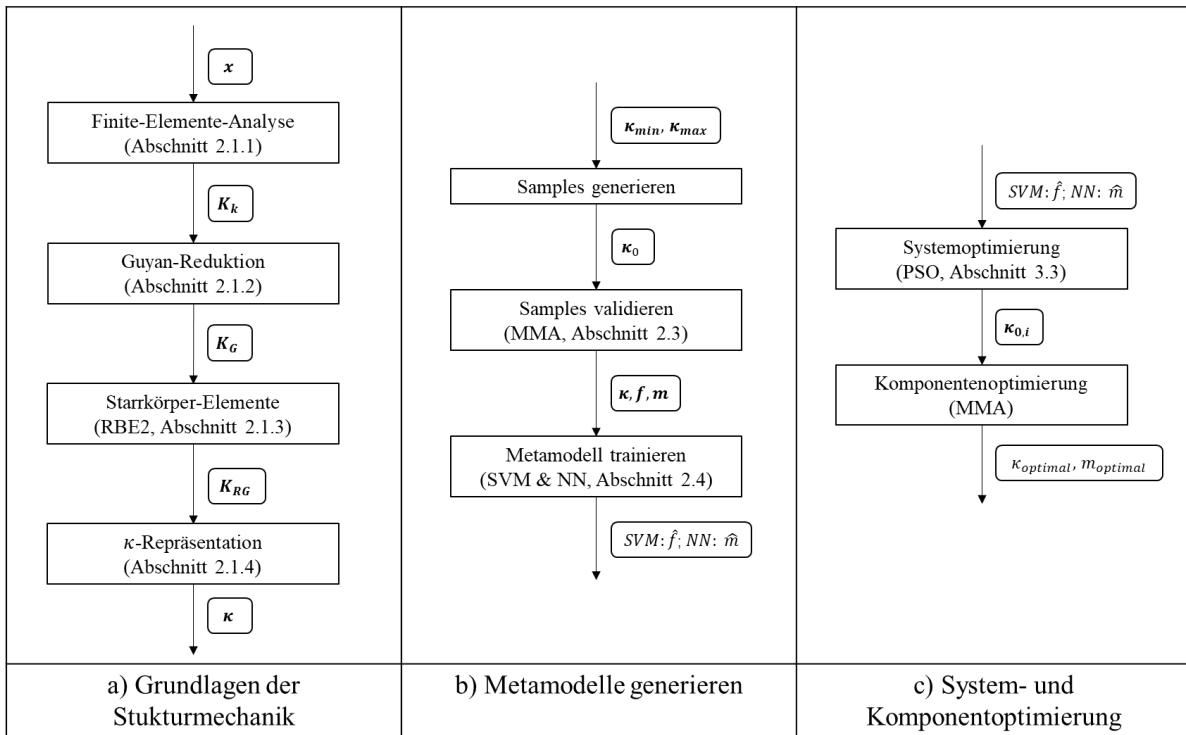


Abb. 1-2: Überblick über Informed Decomposition, bestehend aus: a) der Ausrechnung von  $\kappa$ -Vektoren der Designvariable  $x$ , b) der Erstellung der Metamodelle, und c) der System- und Komponentenoptimierung mit Hilfe der Metamodelle

Abbildung 1-2 zeigt die allgemeine Struktur von Informed Decomposition. Bei c) handelt es sich um die monolithische Optimierung, wobei die Systemoptimierung in c) zum Zweck der verteilten Optimierung eingeführt wird. Um eine vollständige Entkopplung zu erreichen, wird der  $\kappa$ -Vektor in a) eingeführt und durch die Designvariable  $x$  berechnet. Der  $\kappa$ -Vektor ersetzt die Designvariable  $x$ , um das System zu beschreiben. Zur Vermeidung eines Loss of Solution Space werden die Metamodelle in b) eingeführt. Im Wertebereich  $[\kappa_{min}, \kappa_{max}]$  können die Samples zulässig generiert und anschließend für das Training der Metamodelle benutzt werden. Dabei ist zu beachten, dass sich zwar der  $\kappa$ -Vektor aus der Designvariablen  $x$  ableiten lässt, die zulässige Designvariable  $x$  aber nicht unbedingt aus dem zufällig generierten  $\kappa$ -Vektor abgeleitet werden kann. Deshalb wird in Abbildung 1-2-b) der Schritt „Samples validieren“ hinzugefügt, um zu überprüfen, ob der zufällig erzeugte  $\kappa$ -Vektor zulässig ist.

Krischer & Zimmermann (2021) führten den Informed-Decomposition-Ansatz zur Optimierung eindimensionaler Balken und zweidimensionaler Körper mit zwei Freiheitsgraden und zwei Interfaces ein. Kerscher (2021) nutzt diese Methode auch zur Optimierung dreidimensionaler Strukturen mit zwei Freiheitsgraden und zwei Interfaces. Krischer et al. (2022) integrierten zusätzlich *Active-Learning-Strategies*, um die Rechengeschwindigkeit und die Genauigkeit des Ansatzes zu verbessern.

Auf der Grundlage der oben genannten Forschungen wird in dieser Arbeit die Topologieoptimierung ein-, zwei- und dreidimensionaler Strukturen mit drei Freiheitsgraden und zwei Interfaces unter Verwendung von Informed Decomposition durchgeführt. Die Verteilung der Freiheitsgrade ist in folgender Abbildung dargestellt:

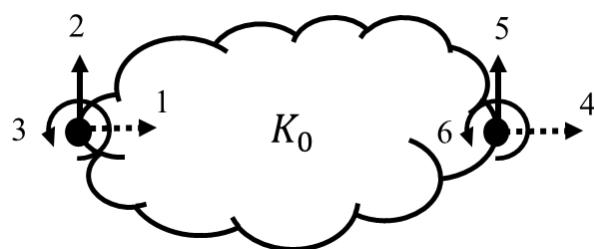


Abb. 1-3: Die zu optimierende Komponente und die Verteilung der Freiheitsgrade. Ausgehend von der ursprünglichen Verteilung mit je zwei Freiheitsgraden und Interfaces (Querkraft 2&5 und Rotation 3&6), werden in dieser Arbeit zwei zusätzliche Freiheitsgrade auf beiden Interfaces eingeführt (durch die gestrichelten Pfeile gekennzeichnet, Längskraft 1&4).

### 1.3 Zielsetzung der Arbeit

Mit einem ähnlichen Verfahren wie Kerscher (2021) werden auf der Grundlage dreier typischer Strukturen (jeweils in einer, zwei und drei Dimensionen) Samples bzw.  $\kappa$ -Vektoren von Designvariablen  $x$  erzeugt und es wird die entsprechende Zulässigkeit  $f$  sowie die Gesamtmasse  $m$  berechnet. Anhand der Zulässigkeits- und Massedatensätze werden mehrere Metamodelle trainiert, wobei für die Zulässigkeitsdatensätze die Klassifikationsmodelle und für die Massedatensätze die Regressionsmodelle trainiert werden. Die leistungsfähigsten Modelle werden für den nächsten Schritt, die Systemoptimierung, ausgewählt. Diese lässt

---

sich mit Hilfe der Metamodelle durchführen, wobei der Zweck darin besteht, das gesamte System zu entkoppeln und jeder Komponente des Systems eine angenäherte Steifigkeitsmatrix zuzuweisen. Anschließend kann jede Komponente separat auf Basis der approximierten Steifigkeitsmatrix optimiert werden. Die Ergebnisse dieser Arbeit werden mit denen der monolithischen Optimierung verglichen, um die Leistung der Informed Decomposition in der strukturierten Optimierung mit drei Freiheitsgraden und zwei Interfaces zu bewerten.

Um den Zeitaufwand für den Forschungsprozess sowie den Rechenaufwand zu verringern, wird ein schrittweises Vorgehen gewählt. Zunächst wird ein eindimensionaler Balken optimiert und während des Debugging-Prozesses werden kontinuierlich Erfahrungen gesammelt, um die am besten geeigneten Optimierungsparameter und Algorithmen zu finden, die im Anschluss direkt in die zweidimensionale Strukturoptimierung einfließen. Derselbe Prozess wird durchgeführt, wenn von zwei auf drei Dimensionen übergegangen wird.

## **1.4 Struktur der Arbeit**

Im folgenden Kapitel werden alle in dieser Arbeit verwendeten mathematischen Methoden und die allgemeine Idee hinter der Informed Decomposition vorgestellt, damit die Leser die Methodik besser nachvollziehen können. Daraufhin wird die allgemeine Methodik auf der Grundlage der Informed Decomposition vorgenommen, die in Kapitel 3 vorgestellt werden. In den Kapiteln 4, 5 und 6 wird die Methode in der ein-, zwei- bzw. dreidimensionalen Strukturoptimierung vorgestellt, in Kapitel 7 erfolgt eine Diskussion der Vor- und Nachteile der Methode. Abschließend wird in Kapitel 8 die Arbeit zusammengefasst und ein Ausblick gegeben.

## 2 Stand der Wissenschaft und Technik

In diesem Kapitel wird das gesamte mathematische und technische Wissen, das für diese Arbeit benötigt wird, vorgestellt. Zunächst wird das Wissen über die Finite-Elemente-Analyse und eine Reihe von Optimierungsmethoden für die Steifigkeitsmatrix vorgestellt, mit denen die Steifigkeitsmatrix anhand der gegebenen Struktur berechnet und weiter vereinfacht werden kann. Im Anschluss werden einige Methoden des Machine-Learning vorgestellt, mit dessen Hilfe die Abbildungsbeziehungen zwischen den vereinfachten Steifigkeitsmatrizen bzw.  $\kappa$ -Vektoren und den ebenfalls durch Sampling gewonnenen Zulässigkeits- und Massendaten hergestellt werden können, die als *Machine-Learning-Modelle*, auch *Metamodelle* genannt, zusammengefasst werden. Am Ende dieses Kapitels werden die Methoden der strukturierten Optimierung zur Komponentenoptimierung vorgestellt.

### 2.1 Grundlagen der Strukturmechanik

In diesem Abschnitt wird das Wissen über den in dieser Arbeit verwendeten Berechnungsteil vorgestellt. Der Hauptprozess ist in Abbildung 1-2-a) dargestellt. Die in dieser Arbeit optimierte mechanische Struktur wird mit 1D-Balken und 2D- bzw. 3D-Topologien realisiert. Durch das Wissen in diesem Abschnitt kann die Steifigkeitsmatrix berechnet und vereinfacht werden.

#### 2.1.1 Finite-Elemente-Analyse (FEM)

Der Grundgedanke der Finite-Elemente-Analyse besteht darin, das zu analysierende System in einzelne Komponenten mit ähnlichen Eigenschaften zu unterteilen und für jede Komponente eine mechanische Analyse nach derselben Methode durchzuführen. Schließlich werden die Ergebnisse der Analyse jeder Einheit summiert, um die Ergebnisse der mechanischen Analyse des gesamten Systems zu erhalten.

Die in dieser Arbeit untersuchte Struktur besteht aus topologischen 1D-, 2D- und 3D-Strukturen, weshalb nur ein Teil der FEM-Methoden, nämlich die statische lineare elastische Theorie, behandelt wird. In diesem Abschnitt wird daher primär das Grundlagenwissen in diesem Teil vorgestellt. Sofern nicht anders angegeben, sind die im Folgenden dargestellten Fachkenntnisse Hughes (2000) und Zienkiewicz (2013) entnommen.



Abb. 2-1: Eine einzelne Finite-Elemente-Komponente mit drei Freiheitsgraden und zwei Interfaces

Für einen untrennbaren Balken (siehe Abbildung 2-1) lautet die Formel der Steifigkeitsgleichung wie folgt:

$$k u = f \quad (2 - 1)$$

$f$  ist der Kraftvektor und besteht in diesem Falle aus sechs Elementen, nämlich den Kräften in Richtung der sechs Freiheitsgrade. Der Verschiebungsvektor  $u$  beschreibt die Verschiebung in Richtung der sechs Grade und  $k$  ist die Steifigkeitsmatrix der Struktur mit einer Größe von  $6 \times 6$ , die die Verformungsbeständigkeit bzw. Steifigkeit basierend auf den gegebenen Kräften beschreibt.

Diese Formel lässt sich, wie unten dargestellt, detaillierter darstellen:

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} \quad (2 - 2)$$

Das einzelne Steifigkeitselement  $k_{ij}$  beschreibt die Steifigkeit anhand der Kraft  $j$  auf der Verschiebung  $i$ .

Wird ein Balken in mehrere Elemente unterteilt (siehe Abbildung 2-2), können für jedes Element Formeln wie (2-1) und (2-2) aufgelistet werden.

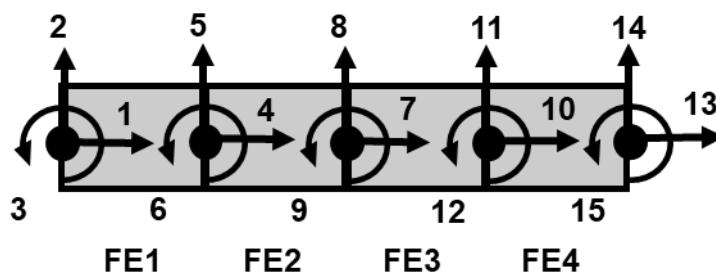


Abb. 2-2: Ein in vier Elemente unterteilter Balken

Die detaillierte Formel von FE2 (Finite Elemente 2) wird wie Gleichung (2-2) geschrieben:

$$\begin{bmatrix} k_{44} & k_{45} & k_{46} & k_{47} & k_{48} & k_{49} \\ k_{54} & k_{55} & k_{56} & k_{57} & k_{58} & k_{59} \\ k_{64} & k_{65} & k_{66} & k_{67} & k_{68} & k_{69} \\ k_{74} & k_{75} & k_{76} & k_{77} & k_{78} & k_{79} \\ k_{84} & k_{85} & k_{86} & k_{87} & k_{88} & k_{89} \\ k_{94} & k_{95} & k_{96} & k_{97} & k_{98} & k_{99} \end{bmatrix} \begin{bmatrix} u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} \quad (2 - 3)$$

Im Vergleich der Gleichungen (2-2) und (2-3) lässt sich feststellen, dass ein Viertel der Steifigkeitsmatrix bei beiden Gleichungen identisch ist, weshalb Letztere kombinierbar sind. Dasselbe gilt für FE3 und FE4. Im Unterschied zu Gleichung (2-1), die die Steifigkeit eines

einzelnen Elements beschreibt, lautet die kombinierte Steifigkeitsgleichung für die Struktur von Abbildung 2-2 wie folgt:

$$K U = F \quad (2 - 4)$$

Der Kraftvektor  $F$  und der Verschiebungsvektor  $U$  bestehen in diesem Fall aus je 15 Elementen, dementsprechend hat die Steifigkeitsmatrix  $K$  eine Größe von  $15 \times 15$ . Die zugehörige Struktur sieht wie folgt aus:

$$K_k = \begin{bmatrix} k_{1 \sim 3,1 \sim 3} & k_{1 \sim 3,4 \sim 6} & 0 & & \\ k_{4 \sim 6,1 \sim 3} & k_{4 \sim 6,4 \sim 6} & k_{4 \sim 6,7 \sim 9} & \cdots & 0 \\ 0 & k_{7 \sim 9,4 \sim 6} & k_{7 \sim 9,7 \sim 9} & & \\ \vdots & & & \ddots & \vdots \\ 0 & & & \cdots & k_{13 \sim 15,13 \sim 15} \end{bmatrix} \quad (2 - 5)$$

Die kombinierte Matrix  $K_k$  ist dünn besetzt und verfügt nur in der Nähe der Diagonalen von oben links nach unten rechts über Werte. Deshalb kann die kombinierte Steifigkeitsmatrix  $K_k$  während der Programmierung in der Software *Matlab* durch den Befehl ‚`sparse()`‘ dargestellt werden, um Rechenzeit zu sparen.

Abbildung 2-2 beschreibt nur die eindimensionale Finite-Elemente-Analyse, jedoch werden in dieser Arbeit auch zwei- und dreidimensionale Analysen betrachtet. Die entsprechende Steifigkeitsmatrix kann von erheblicher Größe sein, aber die Grundstruktur der Steifigkeitsmatrix sowie der Ableitungsprozess ähneln der obigen Darstellung. Die spezifischen Werte der Matrix werden bei ihrer Verwendung in den späteren Kapiteln vorgestellt.

Im Folgenden werden die drei Haupteigenschaften von Steifigkeitsmatrizen erklärt:

1. Symmetrie, d. h.  $K = K^T$  oder  $k_{ij} = k_{ji}$ . Die Bedeutung dieser Eigenschaft liegt darin, dass bei der Definition oder Berechnung einer Steifigkeitsmatrix nur die Hälfte der Steifigkeitselemente berücksichtigt werden muss und die andere Hälfte aufgrund der Symmetrie kopiert werden kann, was den Rechenaufwand reduziert und Zeit spart.
2. Positive- oder semidefinite quadratische Form, d.h.  $x K x^T \geq 0$ . Der Steifigkeitswert ist der Quotient aus der Kraft und der Verschiebung. Da beide Werte positiv oder 0 sind, muss auch der Steifigkeitswert positiv oder 0 sein. Ob eine Steifigkeitsmatrix diese Eigenschaft erfüllt, lässt sich durch Berechnung des Eigenwerts beurteilen. Damit die Eigenschaft als erfüllt gilt, muss der Eigenwert der Steifigkeitsmatrix größer oder gleich Null sein. Diese Eigenschaft kann verwendet werden, um die Zulässigkeit der Steifigkeitsmatrix zu beurteilen.
3. Singularität (Liu & Quek, 2013, S. 91). Entsprechend der *Rigid-Body-Modes* kann sich eine Struktur ohne definierte Unterstützung und ohne Berücksichtigung dynamischer Effekte frei im Raum bewegen. Mathematisch ausgedrückt führt dies dazu, dass Lösungen der Gleichung  $K U = 0$  mit  $U \neq 0$  existieren. Um dieses Problem zu beheben, müssen geeignete Randbedingungen definiert werden. In dieser Arbeit ist es notwendig, eine Singularitätsanalyse der erhaltenen Steifigkeitsmatrix durchzuführen, um sicherzustellen, dass das richtige Ergebnis erhalten wird.

Weitere Details können Hughes (2000), Zienkiewicz (2013) sowie Rust (2016) entnommen werden.

### 2.1.2 Guyan-Reduktion

Wie in Gleichung (2-5) gezeigt, existieren bereits in der kombinierten Steifigkeitsmatrix  $K_k$  bei eindimensionaler Finite-Elemente-Analyse mit vier Finite-Elementen insgesamt 225 Elemente. Eine solche Steifigkeitsmatrix kann nicht direkt zum Trainieren des Metamodells verwendet werden, da eine derartige Vielzahl an Eingabeparametern zu einer zu hohen Rechen- und Modellkomplexität führen kann. Entsprechend der in Abschnitt 2.1.1 vorgestellten Eigenschaften der Steifigkeitsmatrix kann bei Letzterer eine Reihe vereinfachter Methoden angewandt werden. In diesem Abschnitt wird die erste vereinfachte Methode, die sogenannte *Guyan-Reduktion*, die von Guyan (1965) entwickelt wurde, vorgestellt. In einigen Fällen wird auch die Bezeichnung *Irons-Guyan-Reduktion* verwendet, da Irons (1963, 1965) zur gleichen Zeit eine ähnliche Methode entwickelte (J.-G. Kim & Lee, 2014). Mit dieser Methode kann die kombinierte Steifigkeitsmatrix für Balken mit drei Freiheitsgraden und zwei Interfaces zu einer kleineren Steifigkeitsmatrix der Größe  $6 \times 6$  vereinfacht werden.

Ein Blick zurück auf Abbildung 2-2 zeigt, dass nicht alle Knoten direkt mit den äußeren Kräften verbunden sind, sondern nur die sechs Freiheitsgrade auf beiden Seiten der Struktur, d. h. die Knoten 1, 2, 3 sowie 13, 14 und 15. Der Kerngedanke der Guyan-Reduktion besteht darin, durch Vereinfachung der mathematischen Steifigkeitsgleichungen alle Elemente der kombinierten Steifigkeitsmatrix zu eliminieren, die den Knoten ohne äußere Kräfte oder Zwänge entsprechen. Die Knoten ohne externen Kräften oder Zwängen werden als *Slave-Knoten* mit dem Index  $s$  bezeichnet. Die von externen Bedingungen beeinflussten Knoten werden hingegen als *Master-Knoten* (Index  $m$ ) bezeichnet. Durch Rekonstruktion der Gleichungen (2-4) und (2-5) erhält man folgendes Gleichungssystem:

$$\begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \begin{bmatrix} U_m \\ U_s \end{bmatrix} = \begin{bmatrix} F_m \\ F_s \end{bmatrix} \quad (2 - 6)$$

Wie oben erwähnt, wirkt auf Slave-Knoten keine externe Kraft, d. h.  $F_s = 0$ . Daraufhin kann  $U_s$  mit Hilfe der zweiten Gleichung berechnet werden:

$$U_s = -K_{ss}^{-1} K_{sm} U_m \quad (2 - 7)$$

Wird Gleichung (2-7) zurück in die erste Gleichung von (2-6) eingesetzt, wird folgende Gleichung erhalten:

$$(K_{mm} - K_{ms} K_{ss}^{-1} K_{sm}) U_m = F_m \quad (2 - 8)$$

Die Gleichung in Klammern ist als vereinfachte Steifigkeitsmatrix  $K_G$  ( $G$  für Guyan) definiert;

$$K_G = (K_{mm} - K_{ms} K_{ss}^{-1} K_{sm}) \quad (2 - 9)$$

die neue Steifigkeitsgleichung ist unten dargestellt:

$$K_G U_m = F_m \quad (2 - 10)$$

Da der Index  $m$  hier nur sechs Freiheitsgrade enthält, beträgt die Größe der neuen Steifigkeitsmatrix  $K_G$   $6 \times 6$ .

Aufgrund dessen, dass die Gesamtheit des FHG (FHG für Freiheitsgrad) zur Bildung der reduzierten Steifigkeitsmatrix  $K_G$  beiträgt, gibt es keinen Genauigkeitsverlust für die Verschiebungen der Master-FHG (Guyan, 1965).

Die Guyan-Reduktion bietet eine weitere Lösungsmethode, die darin besteht, eine Beziehungsmatrix  $T_G$  zwischen Master- und Slave-Verschiebung zu erstellen (J.-G. Kim & Lee, 2014):

$$U_m = T_G U_s, \quad \text{mit } T_G = \begin{bmatrix} I \\ -K_{ss}^{-1} K_{sm} \end{bmatrix} \quad (2-11)$$

Bei der Anwendung von  $T_G$  kann  $K_G$  wie folgt ausgedrückt werden:

$$K_G = T_G^T K T_G, \quad \text{mit } K = \begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \quad (2-12)$$

Beide Ausdrücke für die reduzierte Steifigkeitsmatrix  $K_G$ , Gleichung (2-9) und (2-12), werden in dieser Arbeit verwendet.

### 2.1.3 Starrkörper-Elemente

Beim eindimensionalen Balken reicht die Guyan-Reduktion aus, um die kombinierte Steifigkeitsmatrix  $K_k$  auf die Größe  $6 \times 6$  zu reduzieren. Bei zwei- und dreidimensionalen Topologien existiert jedoch nicht nur ein Knoten je Ende wie in Abbildung 2-2, sondern es gibt mehrere Knoten pro Seitenquerschnitt (siehe Abbildung 2-3, Knoten  $x_1, x_2, x_3, x_4$  und  $x_{n-3}, x_{n-2}, x_{n-1}, x_n$ ). Das bedeutet, die Anzahl der Masterknoten ist nicht mehr  $2 \times 1 \times 3 = 6$  (zwei Enden, ein Knoten an jedem Ende und drei Freiheitsgrade pro Knoten), sondern  $2 \times n \times 3 = 6n$  ( $n$  bedeutet, dass es an jedem Ende mehrere Knoten gibt). In Gleichung (2-10) besitzt  $K_G$  somit nicht mehr die Größe  $6 \times 6$ , sondern ist komplexer. Es bedarf also einer zusätzlichen Methode, um die Steifigkeitsmatrix weiter zu vereinfachen. Diese Methode wird als *Starrkörper-Elemente* bezeichnet, auf Englisch *rigid body elements*, abgekürzt *RBE2*.

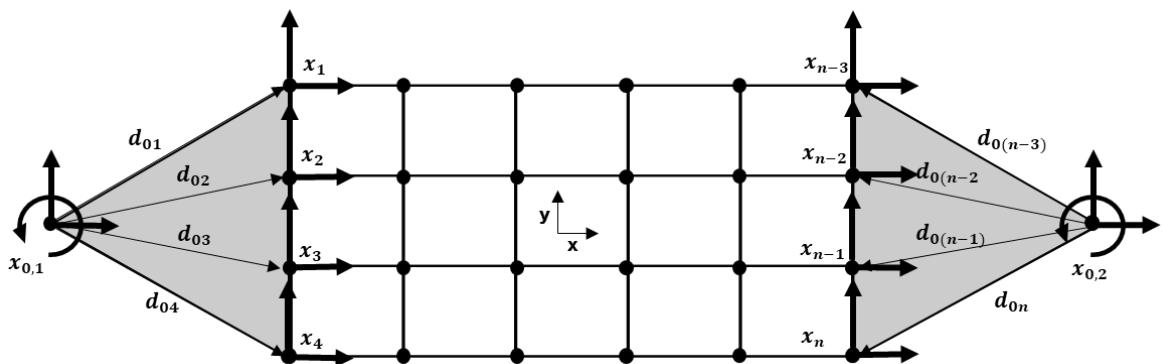


Abb. 2-3: Ein Balken mit der zweidimensionalen Finite-Elemente-Analyse. An beiden Enden werden Starrkörper angenommen. Zwei unabhängige Knoten  $x_{0,1}$  und  $x_{0,2}$  sind mit den entsprechenden abhängigen Knoten  $x_1, x_2, x_3, x_4$  sowie  $x_{n-3}, x_{n-2}, x_{n-1}, x_n$  starr verbunden.

RBE2 wird verwendet, um die Freiheitsgrade wieder auf einen Interfacepunkt zu reduzieren. Zu jedem Seitenquerschnitt wird ein zusätzlicher Knoten  $x_0$  bzw. ein *unabhängiger Knoten* angenommen. Knoten, die auf demselben Seitenquerschnitt wie  $x_0$  (z. B.  $x_1, x_2, x_3, x_4$  für  $x_{0,1}$  und  $x_{n-3}, x_{n-2}, x_{n-1}, x_n$  für  $x_{0,2}$ ) liegen, werden als *abhängige Knoten* bezeichnet. Es

wird davon ausgegangen, dass zwischen den unabhängigen und den abhängigen Knoten starre Verbindungen bestehen, das heißt, die in Abbildung 2-3 schattierten Teile sind starre Körper, wovon sich auch der Name *Starrkörper-Elemente* ableitet. Aufgrund der obigen Bedingungen kann die Bewegung aller abhängigen Knoten durch die des unabhängigen Knotens  $x_0$  und durch eine zusätzliche Umrechnungsgleichung

$$x_i = x_0 + r_0 \times d_{0i}, \quad \text{mit } i = 1, 2, 3, 4, n-3, n-2, n-1, n \quad (2 - 13)$$

beschrieben werden, sodass die Steifigkeitsmatrix der gesamten Topologie durch die Kräfte und Verschiebungen beider unabhängiger Knoten  $x_0$  berechnet werden kann. In der Umrechnungsgleichung sind  $x_0$  die unabhängigen Knoten,  $x_i$  beschreibt die abhängigen Knoten;  $r_0$  ist die Rotation-FHG der unabhängigen Knoten und  $d_{0i}$  ist der Distanzvektor zwischen abhängigen und unabhängigen Knoten. Der Operator  $\times$  bezeichnet das Kreuzprodukt zweier Vektoren. Durch die Kombination der Guyan-Reduktion mit RBE2 lässt sich die Größe der kombinierten Steifigkeitsmatrix von 2D- und 3D-Topologie auf  $6 \times 6$  reduzieren.

Es ist von zentraler Bedeutung, zu erkennen, dass die Einführung eines RBE2-Elements in das Modell die Steifigkeit der Struktur künstlich erhöht, indem einige Verformungen eingeschränkt werden, die in der Realität zulässig wären (Heirman & Desmet, 2010). Die Implementierung einer solchen Starrkörperverbindung kann durch die Konstruktion einer Mehrpunktbedingung gemäß den Anweisungen in Liu und Quek (2013, Abschnitt 11.11) erfolgen.

## 2.1.4 $\kappa$ -Representation

Mit Hilfe der in den vorangegangenen Abschnitten erläuterten Vereinfachungsmethoden kann die kombinierte Steifigkeitsmatrix  $K_k$  in ein-, zwei- und dreidimensionalen Topologien auf eine Steifigkeitsmatrix mit der Größe  $6 \times 6$  vereinfacht werden. Diese Matrix enthält 36 Parameter, die jedoch nicht unabhängig voneinander sind. Diese Matrix sind als die Eingabe vom Metamodell auch nicht klein genug. Daher ist eine weitere, als  $\kappa$ -Repräsentation bezeichnete Vereinfachungsmethode erforderlich. Mit dieser kann die Steifigkeitsmatrix der Größe  $6 \times 6$  mit nur vier Parametern dargestellt werden.

Eine ähnliche Idee wurde von Krischer und Zimmermann (2021) entwickelt, die nur drei Parameter verwenden, um eine Steifigkeitsmatrix mit einer Größe von  $4 \times 4$  für jeweils zwei Freiheitsgraden und Interfaces zu erhalten. Diese Idee wird in der vorliegenden Arbeit auf die  $6 \times 6$ -Steifigkeitsmatrix für drei Freiheitsgrade und zwei Interfaces erweitert, wonach diese Matrix auf vier Parameter vereinfacht werden kann. Bisher lässt sich eine kombinierte Steifigkeitsmatrix, die durch eine Finite-Elemente-Analyse erhalten wurde, durch eine Reihe von Methoden auf vier Parameter reduzieren, die klein genug sind, um als Input für das Metamodell verwendet werden zu können.

Die  $6 \times 6$ -Steifigkeitsmatrix wird hierbei als  $K_0$  bezeichnet; die vier Parameter stellen das erste, zweite bzw. vierte diagonale Element von  $K_0$  dar:

$$\kappa = [K_{0,11}, K_{0,22}, K_{0,33}, K_{0,66}] \quad (2 - 14)$$

Der aus diesen Parametern zusammengesetzte Vektor wird als  $\kappa$ -Vektor bezeichnet, woher auch der Name  $\kappa$ -Repräsentation stammt.

Die übrigen Parameter von  $K_0$  können durch die Parameter in  $\kappa$ -Vektor ausgerechnet werden. Die detaillierten Gleichungen können Anhang 2 entnommen werden.

## 2.2 Machine learning

Nach der Vereinfachung der Steifigkeitsmatrix  $K_0$  besteht der nächste Schritt darin, das Metamodell zu trainieren. Das in dieser Arbeit verwendete Modell besteht aus zwei des Machine-Learning-Modellen. In folgenden Abschnitt werden die verwandten Inhalte des Machine-Learning vorgestellt.

Machine-Learning (ML) ist die Untersuchung von Computeralgorithmen, die sich durch Erfahrung und die Verwendung von Daten automatisch verbessern (Mitchell, 1997). Bei technischen Problemen ist es unter Umständen eine komplizierte oder unmögliche Aufgabe, die interne Struktur eines Systems zu bestimmen oder das System durch mathematische Gleichungen aus dem Ingenieurwissen zu erklären. In diesem Fall stellt Machine-Learning eine Alternative dar. Es kann vorhandene In- und Outputdatensätze verwenden, um eine Abbildungsbeziehung zwischen Eingabe und Ausgabe zu finden. Für Gleichungen, deren interne Strukturen komplex und schwer lösbar sind, kann Machine-Learning eingesetzt werden, um ein Modell zu finden, das diese Gleichung annähert, sodass mit derselben Eingabe in die Gleichung durch das Machine-Learning-Modell eine annähernd gleiche Ausgabe erzielbar ist. In der Praxis kann es sich als effektiver erweisen, der Maschine bei der Entwicklung ihres eigenen Algorithmus zu helfen, anstatt jeden erforderlichen Schritt von menschlichen Programmierern spezifizieren zu lassen (Ethem, 2020).

Die beiden Hauptkategorien des Machine-Learnings sind das überwachte und das unüberwachte Lernen. Algorithmen des überwachten Lernens erstellen ein mathematisches Modell eines Datensatzes, der sowohl die Eingaben als auch die gewünschten Ausgaben enthält (Russell & Norvig, 2010). Diese Daten werden als *Trainingsdaten* bezeichnet und bestehen aus einem Satz von Trainingsbeispielen. Jedes Trainingsbeispiel hat eine oder mehrere Eingaben sowie eine gewünschte Ausgabe, die auch als *Überwachungssignal* bezeichnet wird. Durch iterative Optimierung einer Zielfunktion unter Nutzung der Überwachungssignale lernen Algorithmen des überwachten Lernens eine Funktion, die zur Vorhersage der mit neuen Eingaben verbundenen Ausgabe verwendet werden kann (Mohri et al., 2012). Zu den überwachten Lernalgorithmen gehören aktives Lernen, Klassifizierung und Regression (Alpaydin, 2010). Klassifizierungsalgorithmen werden verwendet, wenn die Ausgaben auf eine begrenzte Anzahl von Werten beschränkt oder, einfach ausgedrückt, „diskret“ sind. Regressionsalgorithmen finden Anwendung, wenn die Ausgaben jeden beliebigen numerischen Wert innerhalb eines Bereichs haben können bzw. „kontinuierlich“ sind. Ähnlichkeitslernen ist ein Bereich des überwachten maschinellen Lernens, der eng mit Regression und Klassifizierung verwandt ist. Allerdings besteht das Ziel darin, aus Beispielen zu lernen, indem eine Ähnlichkeitsfunktion verwendet wird, die misst, wie ähnlich oder verwandt zwei Objekte sind. Unüberwachte Lernalgorithmen verwenden

einen Datensatz, der nur Eingaben enthält, und suchen nach einer Struktur in den Daten, z. B. durch Gruppieren oder Clustern von Datenpunkten. Die Algorithmen lernen daher von Testdaten, die nicht gekennzeichnet, klassifiziert oder kategorisiert wurden.

Die beiden in dieser Arbeit verwendeten Methoden sind SVM (Support Vector Machine) aus dem Bereich der Klassifizierung und NN (Neuronales Netz) aus dem Bereich der Regression im Rahmen des überwachten Lernens, weshalb in diesem Abschnitt nur diese beiden Methoden vorgestellt werden.

Die beiden in dieser Arbeit verwendeten Methoden sind SVM (Support Vector Machine) aus dem Bereich der Klassifizierung und NN (Neuronales Netz) aus dem Bereich der Regression im Rahmen des überwachten Lernens, so dass in diesem Abschnitt hauptsächlich nur diese beiden Methoden vorgestellt werden.

### 2.2.1 Support-Vector-Machine

SVM ist ein weitverbreitetes Klassifikationsverfahren, das von Boser et al. (1992) entwickelt wurde. SVM hat Anerkennung für ihre Fähigkeit erlangt, mit hoher Wahrscheinlichkeit die richtige Entscheidung zu treffen und eine Vielzahl unterschiedlicher und möglicherweise hochdimensionaler Daten zu verarbeiten.

In der Matlab-Software gibt es bereits die gepackte Toolbox über SVM, daher wird hier hauptsächlich die Idee der SVM anstelle der Formelableitung vorgestellt. Sofern nicht ausdrücklich angegeben, stammt der Inhalt dieses Abschnitts aus Hastie et al. (2017, Kapitel 4 und 12).

SVM wird primär zur Lösung binärer Klassifizierungsprobleme verwendet, d. h. die Ausgabe von SVM verfügt nur über zwei Möglichkeiten. SVM wurde aus der von Cortes und Vapnik (1995) vorgeschlagenen *Optimal Separating Hyperplane* entwickelt, ist robuster gegenüber Datensätzen mit überlappenden Regionen und kann zur Lösung binärer Klassifizierungsprobleme verwendet werden, die nicht linear trennbar sind. Um den Einstieg in das Konzept der SVM zu erleichtern, beginnt dieser Abschnitt mit Optimal Separating Hyperplane. Damit die Abbildungen leicht verständlich sind, wurden für diesen Abschnitt zweidimensionale Darstellungen gewählt. Die SVM-Methode ist jedoch auch im mehrdimensionalen Raum anwendbar.

Wenn Lösungen für binäre Klassifizierungsprobleme existieren, sind es im Allgemeinen unendlich viele, wie in der folgenden Abbildung gezeigt wird:

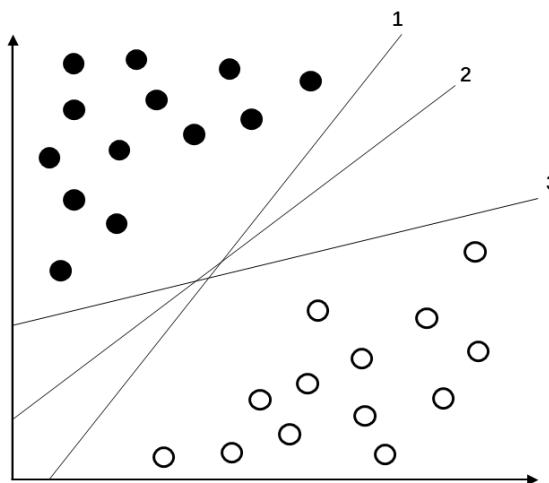


Abb. 2-4: Der ausgefüllte und der hohle Kreis stellen je einen Datensatztyp dar. In dieser Abbildung sind die beiden Datensatztypen linear trennbar. Wird eine Hyperebene (hier als gerade Linie dargestellt) verwendet, um die beiden Datensätze zu trennen, ist es leicht verständlich, dass unendlich viele solcher Hyperebenen existieren. Drei Beispiele sind in der Abbildung dargestellt.

Der Zweck dieser Methode besteht darin, aus einer unendlichen Anzahl von Hyperebenen diejenige zu finden, die für die Leistung des Klassifikators am günstigsten ist, d. h. die Hyperebene, die den *Margin* maximiert. Letzterer stellt den Abstand der Hyperebene am nächsten liegenden Punkte in einer Klasse zur Hyperebene dar, wie in der folgenden Abbildung gezeigt. Der Margin der optimierten Hyperebene zu den beiden Klassen sollte identisch sein.

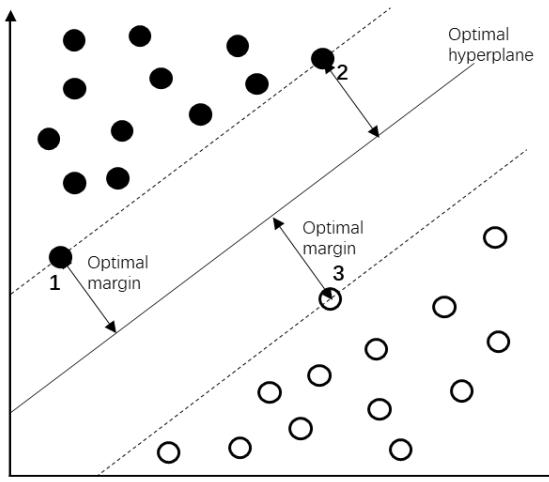


Abb. 2-5: Die optimierte Hyperebene und der entsprechende optimierte Margin. Erstere wird durch eine durchgezogene Linie dargestellt, die beiden Linien parallel zur Hyperebene, die zur Beschreibung des Margins verwendet werden, durch eine gestrichelte. Wie in der Abbildung zu erkennen ist, liegen insgesamt drei Punkte in den beiden Klassen auf der gestrichelten Linie. Die Abstände der Punkte 1, 2 und 3 zur Hyperebene sind die Größen des Margins und sollen identisch sein.

Die Steigung  $\omega$  und der Achsenabschnitt  $b$  der Hyperebene können am Anfang nicht bestimmt werden, d. h. zu Beginn wird eine Hyperebene willkürlich ausgewählt.

Anschließend werden die optimale Steigung und der optimale Achsenabschnitt der Hyperebene durch ein Optimierungsverfahren berechnet. Der Zweck der Optimierung besteht darin, den maximalen Wert des Margins  $M$  zu finden. Die zu erfüllende Bedingung besteht darin, dass der Abstand von jedem Punkt im Datensatz zur Hyperebene größer als der Rand ist. Dieses Optimierungsverfahren gestaltet sich wie folgt:

$$\begin{aligned} & \max_{\omega, b, \|\omega\|=1} M \\ & \text{subject to: } y_i(x_i^T \omega + b) \geq M, \quad i = 1, \dots, N \end{aligned} \quad (2-15)$$

Dabei ist zu beachten, dass der Teil  $(x_i^T \omega + b)$  sich auf den Abstand von einem Punkt  $i$  zur Hyperebene bezieht. Der positive und der negative Wert des Abstands sind zu beachten; sie entsprechen den beiden Seiten der Hyperplane.  $y_i$  bezieht sich auf die Klassen des Punktes  $i$  (als +1 oder -1 definiert). Da das Vorzeichen der Klasse beliebig angegeben werden kann, wird davon ausgegangen, dass das Vorzeichen des Abstands gleich dem Vorzeichen der Klasse ist, wodurch sichergestellt wird, dass das Produkt der beiden größer als 0 ausfällt. Um dieses Optimierungsproblem zu lösen, kann die Lagrange-Gleichung verwendet werden; zum spezifischen Lösungsverfahren siehe Hastie et al. (2017, S. 132–133). Bei dieser Methode bezieht sich die Position der Hyperebene nur auf den Datenpunkt am Rand der Klasse, der der Hyperebene am nächsten liegt, und hat nichts mit den Punkten innerhalb des Datensatzes zu tun, weshalb die Robustheit dieser Methode angemessen ist.

Die Mehrheit der realen Datensätze sieht jedoch nicht wie in Abbildung 2-5 aus. In Anbetracht der folgenden beiden Datensatzverteilungen (Abbildungen 2-6 und 2-7) müssen einige Änderungen an der Gleichung (2-15) vorgenommen werden.

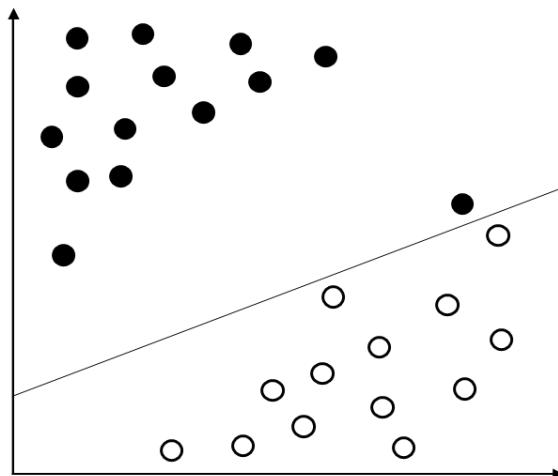


Abb. 2-6: Durch das Hinzufügen eines weiteren Punktes ändert sich die gesamte Hyperebene signifikant. Dieser Punkt ist weit von anderen Punkten derselben Klasse entfernt und nicht repräsentativ.

Mit diesen beiden Fällen (Abbildung 2-6 und 2-7) kann das Optimierungsverfahren durch die Methode *Soft-Margin-Hyperplane* verbessert werden. Dieser Ansatz toleriert die Fehlklassifizierung einiger Datenpunkte und zielt darauf ab, den Großteil der Punkte den beiden Klassen zu klassifizieren.

Auf Basis der Gleichung (2-15) wird eine neue Variable  $\xi_i$  hinzugefügt; für richtig

klassifizierte Punkte  $i$  ist  $\xi_i = 0$ , für falsch klassifizierte Punkte nimmt die Variable einen Wert größer als 0 an, der dem Abstand des Punktes zur entsprechenden Margin-Ebene entspricht. Der mathematische Ausdruck des Optimierungsproblems nach Einführung von  $\xi$  lautet wie folgt:

$$\max_{\omega, b, \|\omega\|=1} M$$

*subject to:*  $y_i(x_i^T \omega + b) \geq M(1 - \xi_i), \quad i = 1, \dots, N$

$$\xi_i \geq 0, \quad \sum_i \xi_i \leq c \quad (2-16)$$

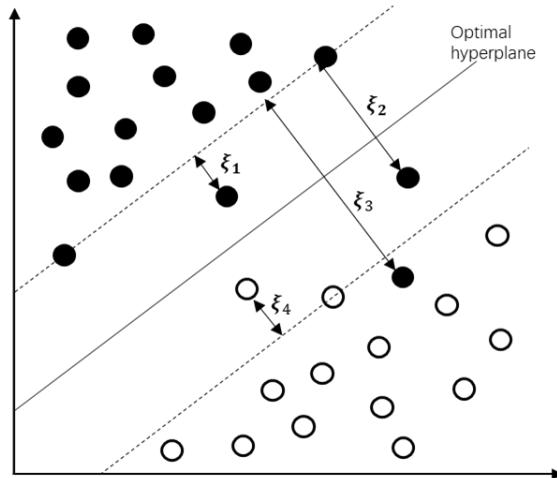


Abb. 2-7: Die beiden Klassen haben sich überlappt und können nicht durch eine Hyperebene unterschieden werden.

Dabei ist  $c$  in Gleichung (2-16) ein künstlich definierter Parameter, der die Anzahl der falsch klassifizierten Daten begrenzt. Die Gleichung (2-16) kann auch durch die Lagrange-Gleichung gelöst werden; zum spezifischen Lösungsverfahren siehe Hastie et al. (2017, S. 418–421).

Darüber hinaus ist das binäre nichtlineare Klassifikationsproblem zu berücksichtigen, wie in Abbildung 2-8 gezeigt. Aus dieser Abbildung ist ersichtlich, dass die beiden Klassen nicht durch eine Hyperebene, sondern nur durch einen Kreis geteilt werden können. Eine angemessene Möglichkeit, dieses Problem zu lösen, stellt die Einführung einer Kernelfunktion dar. Wird beispielsweise eine Gleichung  $z = x^2 + y^2$  eingeführt, sieht die neue Verteilung von  $x$  und  $z$  wie in Abbildung 2-9 aus. Offensichtlich kann eine Hyperebene verwendet werden, um die beiden Klassen zu teilen.

Im Allgemein ist die am häufigsten verwendete Kernelfunktion die radiale Basisfunktion, die auch in dieser Arbeit verwendet wird; zum spezifischen Lösungsverfahren hierbei siehe Hastie et al. (2017, S. 423–424).

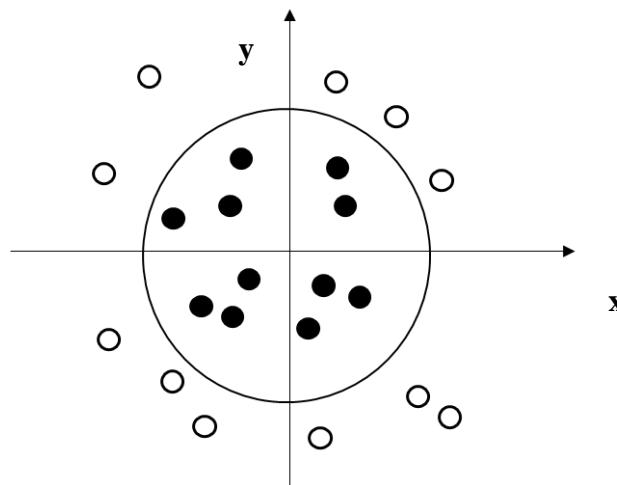


Abb. 2-8: Ein binäres nichtlineares Klassifikationsproblem, das nicht durch eine 2D-Hyperebene separiert werden kann.

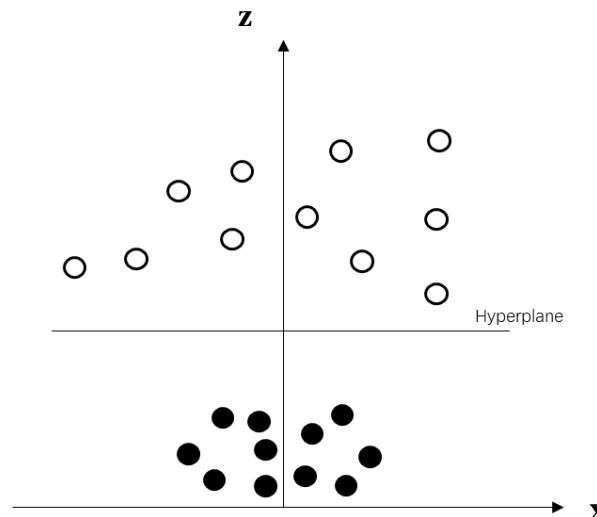


Abb. 2-9: Das durch die Kernelfunktion transformierte nichtlineare Klassifikationsproblem. Dieses kann mittels der Hyperebene separiert werden.

## 2.2.2 Neural Nets

Das neuronale Netzwerk ist ein Machine-Learning-Regressionsmodell. Im Gegensatz zur SVM ist die Ausgabe des neuronalen Netzwerks nicht binär, sondern kann eine kontinuierliche, unendliche Anzahl von Werten darstellen. Theoretisch kann ein neuronales Netzwerk, basierend auf verschiedenen Strukturen (unterschiedliche Anzahl an Layern und Nodes sowie verschiedene Aktivierungsfunktionen), nahezu alle Funktionen simulieren. Es ist deswegen ein leistungsfähiges Verfahren. Da die Toolbox für neuronale Netze auch in Matlab integriert ist, geht es in diesem Abschnitt mehr um die Struktur und die Ableitungsideen des neuronalen Netzes als um den mathematischen Ableitungsprozess. Sofern nicht anders angegeben, stammen die in diesem Abschnitt dargestellten Inhalte aus Schröder (2010).

Zunächst wird die grundlegende Struktur des neuronalen Netzes vorgestellt, einschließlich

Gewichtungen und Aktivierungsfunktionen:

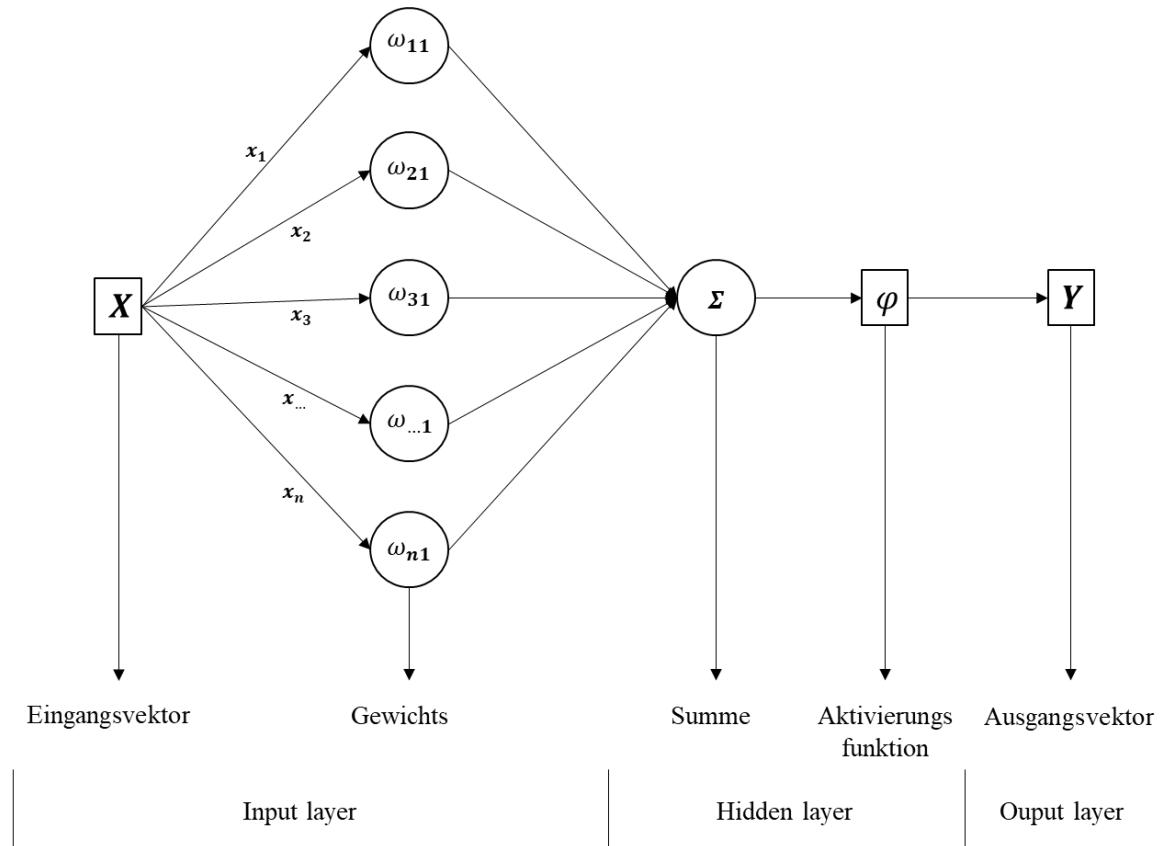


Abb. 2-10: Grundlegende Struktur des neuronalen Netzes mit nur einem Hidden Layer, einem Neuron und einem Ausgang.

Eingangs- und Ausgangsvektorgröße des neuronalen Netzes sollten mit denen des gegebenen Datensatzes identisch sein. Jede Eingabe aus dem Eingabevektor  $X$  beeinflusst die Ausgabe des Systems auf unterschiedliche Weise, was durch die Multiplikation jeder Eingabe mit einem unterschiedlichen Gewicht erreicht wird. Das Ziel beim Training des neuronalen Netzes ist es, einen geeigneten Gewichtsvektor zu finden. Anschließend sollte jeder Eingang mit der Summenfunktion verbunden werden. Die Summe wird durch eine Aktivierungsfunktion mit Schwellenwert überprüft, wobei der überprüfte Wert die Ausgabe des neuronalen Netzes ist. Ein Beispiel für eine Aktivierungsfunktion ist die Treppenfunktion, die hauptsächlich in logischen neuronalen Netzen verwendet wird. Besteht die Summe die Prüfung, ist der Ausgangswert gleich 1. Umgekehrt, d. h. wenn die Summe kleiner als der Aktivierungswert ausfällt, wird der Ausgangswert 0 angegeben. Diese beiden Werte, 0 und 1, entsprechen der falschen und der richtigen Logik. Es existieren zahlreiche Arten von Aktivierungsfunktionen, z. B. die Sigmoid-Funktion,

$$\sigma(v) = \frac{1}{1 + e^v} \quad (2 - 17)$$

die radiale Basisfunktion (RBF)

$$A_j(u) = e^{-\frac{(u-\xi_j)^2}{2 \cdot \sigma^2}} \quad (2-18)$$

und das General-Regression-Neural-Network (GRNN).

$$A_j(u) = \frac{e^{-\frac{(u-\xi_j)^2}{2 \cdot \sigma_{norm}^2 \cdot \Delta \xi^2}}}{\sum_{k=1}^r e^{-\frac{(u-\xi_k)^2}{2 \cdot \sigma_{norm}^2 \cdot \Delta \xi^2}}} \quad (2-19)$$

Die Ausgabe von Abbildung 2-10 kann dann mathematisch wie folgt geschrieben werden:

$$\hat{y} = \varphi(\Sigma(\omega x)) \quad (2-20)$$

Die Werte aller optimierten Gewichte sind vor dem Training des neuronalen Netzmodells unbekannt, sodass davon ausgegangen werden kann, dass alle Gewichte anfangs gleich Null sind, d. h.  $\omega = 0$ . Auf diese Weise können die Eingabewerte des vorhandenen Datensatzes und die gegebene Aktivierungsfunktion verwendet werden, um die geschätzten Ausgaben gemäß Gleichung (2-20) zu erhalten. Diese gleichen zu Beginn mit Sicherheit nicht den korrekten Outputs, die der Datensatz liefert. Die Abweichung zwischen den korrekten Ausgaben  $y$  und den geschätzten Ausgaben  $\hat{y}$  wird als *Ausgangsfehler*  $e$  bezeichnet:

$$e = (y - \hat{y}) \quad (2-21)$$

Um spätere Berechnungen zu erleichtern, dient der folgende, geläufigere Ausdruck für Ausgangsfehler:

$$E = \frac{1}{2} e^2 = \frac{1}{2} (y - \hat{y})^2 \quad (2-22)$$

In den Gleichungen des Ausgangsfehlers (Gleichung (2-22) und Gleichung (2-20)) stellen nur die Gewichte des neuronalen Netzes einen unbekannten Parameter dar. Um den Einfluss jedes Gewichts auf den Ausgangsfehler zu berechnen, kann dieser von jedem Gewicht abgeleitet werden:

$$\Delta \omega_i = -\frac{\partial E}{\partial \omega_i}, \quad i = 1 \dots n \quad (2-23)$$

Diese Ableitung stellt den proportionalen Zusammenhang zwischen dem Ausgangsfehler und einem bestimmten Gewicht dar. Bei einer Änderung des Gewichtswerts  $\omega_i$  ändert sich gleichzeitig der Ausgangsfehler  $E$  proportional zum Ableitungswert. Um das neuronale Netzmodell an das tatsächliche Modell anzupassen, sollte der Ausgangsfehler kontinuierlich reduziert werden, sodass er am Ende möglichst nahe bei 0 liegt. Basierend auf der proportionalen Beziehung zwischen den Gewichten und dem Ausgangsfehler sollten auch die Gewichte kontinuierlich reduziert werden, d. h. die Gewichtsänderung sollte umgekehrt zur Ableitungsrichtung erfolgen, was auch der Grund ist, warum in Gleichung (2-23) ein negatives Vorzeichen steht. Dieses Verfahren wird als *Gradientenabstiegsverfahren* bezeichnet.

Der nächste Schritt besteht darin, alle Gewichte im neuronalen Netz zu erneuern. Da die Erneuerung des Gewichtes rückwärts vom Ausgangsfehler zu den Gewichten verläuft, wird

die Methode dieses Schritts als *Backpropagation* bezeichnet.

$$\omega_i^{(l+1)} = \omega_i^{(l)} + \eta^{(l)} \Delta \omega_i^{(l)} \quad (2 - 24)$$

$l$  und  $l + 1$  stellen die Anzahl der Iterationen dar,  $\eta$  ist eine künstlich definierte, als *Lernparameter* bezeichnete Konstante, die verwendet wird, um die Größe der Gewichtsänderung jeder Erneuerung zu steuern. Ein zu kleiner  $\eta$  erhöht die Anzahl der Iterationen und die Trainingszeit, ein zu großer  $\eta$  kann dazu führen, dass das System nicht konvergieren kann.

Der Vorgang bei den Gleichungen (2-20) bis (2-24) wird mehrmals durchgeführt, bis der Ausgangsfehler klein genug ist. Der Änderungstrend des Ausgangsfehlers und eines Gewichts mit zunehmenden Iterationen ist in der folgenden Abbildung dargestellt.

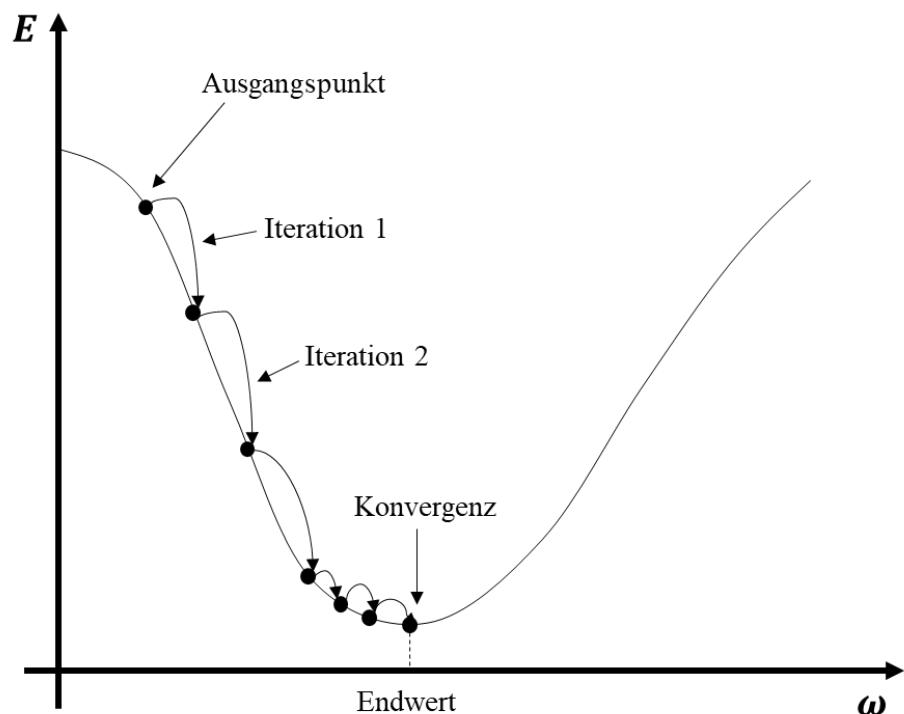


Abb. 2-11: Verhältnis zwischen Ausgangsfehler und Gewicht

Die folgende Abbildung zeigt das Ergebnis der Anpassung mehrerer Daten mittels eines neuronalen Netzes mit einem Hidden Layer und elf Neuronen gemäß GRNN:

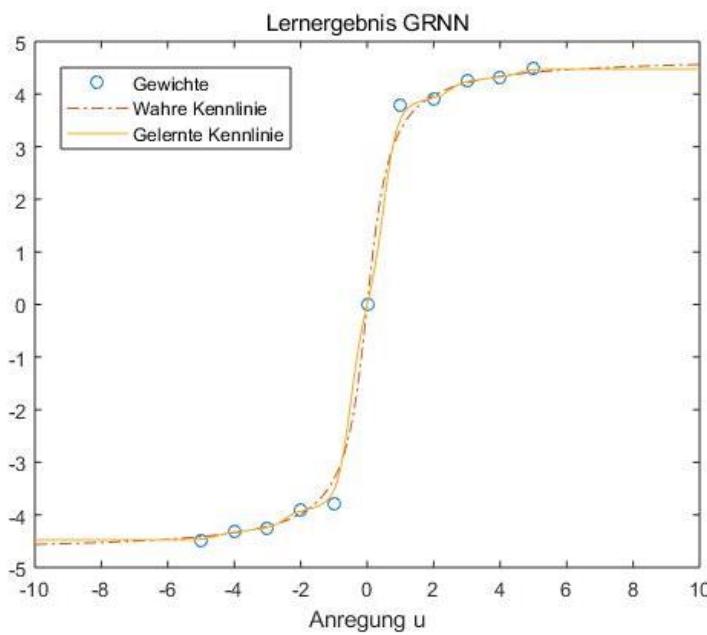


Abb. 2-12: Lernergebnisse von GRNN mit der ursprünglichen Gleichung  $NL(u) = 3 \cdot \arctan(2 \cdot u)$ , darin  $r = 11$  und  $\sigma_{\text{norm}} = 0.45$ .

Am Ende dieses Abschnitts wird Abbildung 2-10 erweitert, um ein allgemeines neuronales Netz zu erhalten, das in folgender Abbildung dargestellt ist:

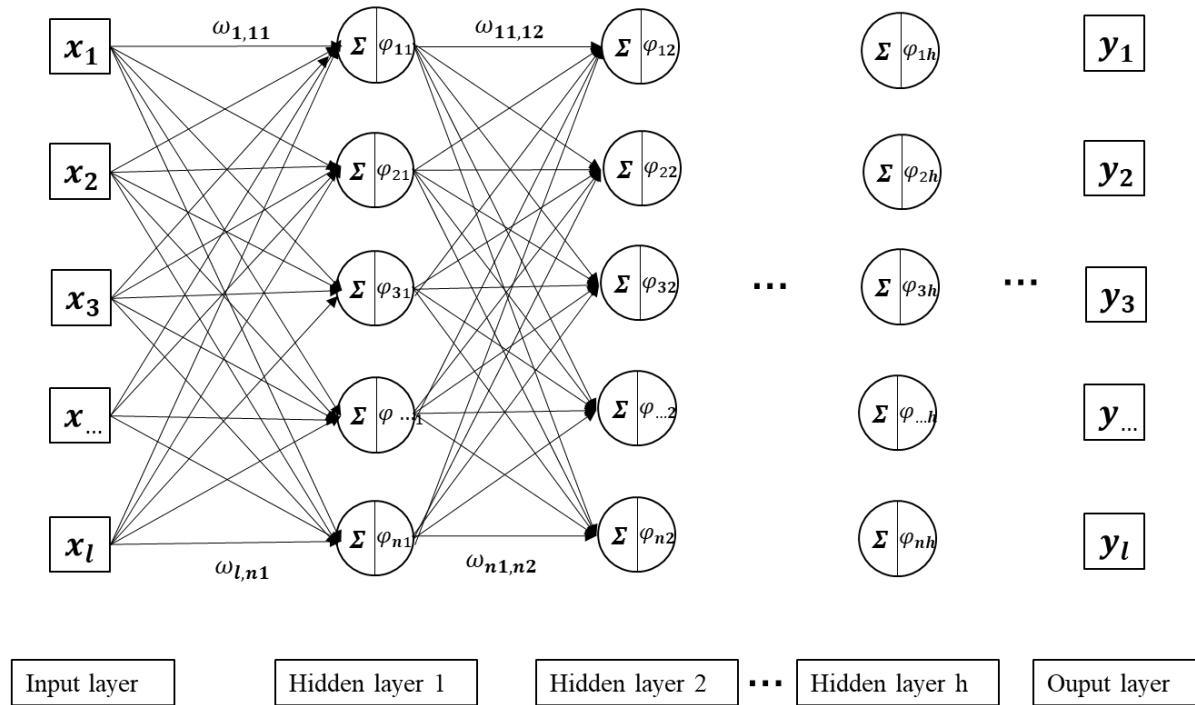


Abb. 2-13: Multilayer-Perceptron (MLP) mit je einer Ein- und Ausgabe,  $h$  Hidden Layern und  $n$  Neuronen je Hidden Layer.

## 2.3 Strukturoptimierung

Der nächste Schritt besteht darin, das via Machine-Learning gewonnene Metamodell zur Strukturoptimierung zu verwenden, weshalb in diesem Abschnitt die Grundlagen des Letzteren vorgestellt werden.

Die Topologie-Optimierung wurde erstmals im Jahr 1988 von Bendsøe und Kikuchi etabliert (Bendsøe & Kikuchi, 1988). Seitdem werden zahlreiche Forschungsarbeiten in dieses Thema gesteckt, was zu vielen verschiedenen Optimierungsverfahren führte. Mit allen diesen Forschungsarbeiten wird dasselbe Ziel verfolgt: mit möglichst wenig Material dieselbe Steifigkeit bzw. mit demselben Material eine höhere Steifigkeit zu erreichen. Bis heute werden Ansätze entwickelt, um Topologie-Optimierungsprobleme zu lösen, z. B. Dichte, Levelset, topologische Ableitung, Phasenfeld, evolutionäre Ansätze etc. (Sigmund & Maute, 2013).

Zunächst muss das Standardproblem der Topologie-Optimierung formuliert werden. Daher wird der betrachtete Bereich durch finite Elemente diskretisiert und der SIMP-Ansatz (Simplified Isotropic Material with Penalization) gemäß der Arbeit von Sigmund (2001) angewendet.

Das grundlegende Optimierungsproblem lautet wie folgt:

$$\begin{aligned} \min_x c(x) &= U^T \cdot K \cdot U = \sum_{e=1}^{n_{ele}} (x_e)^p \cdot u_e^T \cdot k_e \cdot u_e \\ \text{subject to: } &\frac{V(x)}{V_0} = f; K \cdot U = F; 0 < x_{min} < x \leq 1 \end{aligned} \quad (2-25)$$

Die Zielfunktion ist die Compliance  $c$  der Struktur, wobei  $c$  umgekehrt proportional zur Steifigkeit ist. Um die maximale Steifigkeit zu ermitteln, muss der minimale Wert der Compliance eruiert werden. Im mathematischen Ausdruck der Compliance steht  $U$  für die globale Verschiebung und  $K$  für die globale Steifigkeitsmatrix; dementsprechend sind  $u$  und  $k$  die Verschiebung bzw. die Steifigkeitsmatrix eines einzelnen finiten Elements. Die Struktur wird in  $n_{ele}$  finite Elemente  $e$  unterteilt, von denen jedem eine relative Dichte  $x_e$  zugewiesen wird. Um die Singularität der Steifigkeitsmatrix zu vermeiden, muss die relative Dichte  $x_e$  auf etwas mehr als 0 begrenzt werden (in der Gleichung als  $x_{min}$  definiert). Der Bestrafungsfaktor  $p$  wird verwendet, um sogenannte Schwarz-Weiß-Lösungen zu erzwingen, indem Werte  $x_e$ , die bereits nahe 0 sind, weiter in Richtung 0 getrieben werden, wodurch sichergestellt wird, dass kein Material in leeren Bereichen vorhanden ist. Dazu muss der Bestrafungsfaktor  $p > 1$  sein, in der Regel wird  $p = 3$  oder  $4$  gewählt.  $F$  beschreibt die globale Kraft und  $f$  beschreibt der auferlegte Volumenanteil, während  $V(x)$  und  $V_0$  das tatsächliche Material- bzw. das Gesamtdomänenvolumen definieren.

Da das Optimierungsproblem, nämlich die Gleichung (2-25), mit kontinuierlichen Designvariablen formuliert wird, können unter anderem Optimality-Criteria-(OC-)Methoden, Sequential-Linear-Programming-Methoden, die Methode der Moving Asymptotes (MMA) oder effiziente gradientenbasierte Algorithmen angewendet werden, um das Problem zu lösen (Sigmund, 2001).

Die in dieser Arbeit hauptsächlich verwendete Methode ist MMA. Diese wurde ursprünglich im Jahr 1987 von Svanberg vorgeschlagen, um einen stabilen Lösungsalgorithmus für Strukturoptimierungsprobleme bereitzustellen. Da die in Matlab implementierte gekapselte MMA-Funktion in der Literatur (Svanberg, 1987) schon vorgegeben ist, wird hier nur das grundlegende Optimierungsproblem von MMA vorgestellt. Der detaillierte Ableitungsprozess und die Erweiterung von MMA ist in Svanberg (1987) und Svanberg (2002) zu finden.

$$\begin{aligned} \min_x f_0(x) + a_0 \cdot z + \sum_{i=1}^m (c_i \cdot y_i + \frac{1}{2} \cdot d_i \cdot y_i^2) \\ \text{subject to: } f_i(x) - a_i \cdot z - y_i \leq 0, \quad i = 1, \dots, m; \\ x_{j,low} \leq x_j \leq x_{j,upp}, \quad j = 1, \dots, n; \\ y \geq 0, \quad z \geq 0, \end{aligned} \tag{2-26}$$

$f_0(x)$  ist die Zielfunktion des ursprünglichen Optimierungsproblems,  $x$  die ursprüngliche Designvariable. Die Parameter  $y$  und  $z$  sowie die Konstanten  $a_0, a_i, c_i$  und  $d_i$  sind künstlich definiert. Bei Gleichung (2-27) handelt es sich um eine Rekonstruktion des ursprünglichen Optimierungsproblems, um dieses leichter zu lösen. Darin können die Konstanten  $c_i$  sowohl beliebig als auch groß gewählt werden, z. B. 1000 für jedes  $i$ , um einen hohen Einfluss von  $y_i$  auf das Minimierungsproblem zu provozieren.

Um die MMA-Gleichung zu lösen, ist es zudem nötig, die Gradientenfunktion einiger Parameter in die ursprüngliche Optimierungsgleichung einzuführen. Die Funktion wird jedoch erst in den folgenden Kapiteln eingeführt, wenn das Strukturoptimierungsverfahren verwendet wird, damit die detaillierten Gleichungen besser verstanden werden.

## 3 Allgemeine Methodik

In diesem Kapitel werden der allgemeine Ansatz und die Parameter für die strukturierte Topologieoptimierung vorgestellt. In den nachfolgenden Kapiteln werden nur diejenigen Änderungen präsentiert, die vom Inhalt dieses Kapitels abweichen.

### 3.1 Problemstellung

Die in dieser Arbeit analysierte Grundstruktur ist in der folgenden Abbildung dargestellt: ein System mit zwei Trägern, bei dem das linke Ende des linken Trägers mit der Wand verbunden ist und somit ein festes Ende darstellt, während das rechte Ende des rechten Trägers ein freies Ende ist, auf das äußere Kräfte und Verschiebungsbeschränkungen einwirken. Am Verbindungspunkt der beiden Balken sowie am freien Ende gibt es jeweils drei Freiheitsgrade (1, 2 und 3 sowie 4, 5 und 6). Die Länge jeder Komponente beträgt 300 mm. Die beiden Komponenten werden mit  $K1$  bzw.  $K2$  bezeichnet.

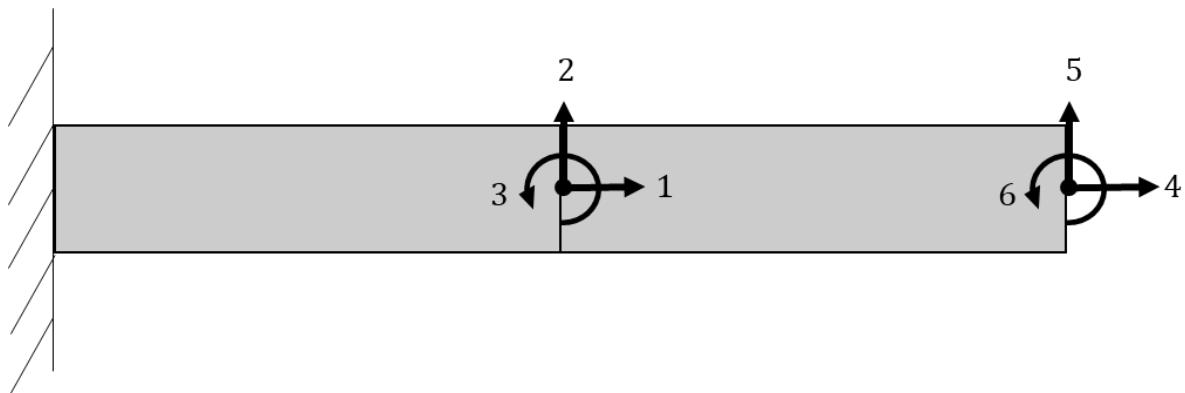


Abb. 3-1: Das Verbindungsverhältnis zwischen den beiden Trägern und der Wand

Spezifische Parameter zu den verwendeten Materialien des Systems können nachfolgender Tabelle entnommen werden:

Material	Elastizitätsmodul $E$	Poisson-Verhältnis $\nu$	Dichte $\rho$
Aluminium	70.0 [GPa]	0.3	2.70 [g/cm <sup>3</sup> ]

Tabelle 3-1: Spezifische Parameter der verwendeten Materialien

### 3.2 Sample-Erzeugung

Wie Abbildung 1-2-a) zeigt, besteht der erste Schritt der Informed Decomposition darin, durch die Designvariable  $x$  den  $\kappa$ -Vektor zu berechnen. Nach diesem Prozess lassen sich die Höchst- und Mindestwerte von  $\kappa$ , nämlich  $\kappa_{min}$  und  $\kappa_{max}$ , durch die Höchst- und Mindestwerte von  $x$  ermitteln. Anschließend kann, wie Abbildung 1-2-b) zeigt, im

Wertebereich  $[\kappa_{min}, \kappa_{max}]$  der  $\kappa$ -Vektor bzw. das Sample zufällig generiert werden.

Wie in Abschnitt 1.2 erwähnt, ist dabei zu beachten, dass nicht jedes Sample im Designraum, d. h. zwischen  $\kappa_{max}$  und  $\kappa_{min}$ , in einer realisierbaren Topologie umgekehrt abgeleitet werden kann, sondern dass nur Samples aus einem kleinen Bereich des Designraums realisierbar bzw. zulässig sind. Der Bereich, in dem alle Samples zulässig sind, wird als *zulässiger Bereich* bezeichnet und der, in dem alle Samples unzulässig sind, als *unzulässiger Bereich*. Bei der Durchführung eines vollständig zufälligen Samplings fallen die meisten Samples in den unzulässigen Bereich, was zu einem signifikanten Unterschied in der Anzahl der zulässigen und unzulässigen Samples führt. Der Beeinfluss von diesem Unterschied ist, dass die Metamodelle, die mit solchen Samples trainiert werden, eine unzureichende Genauigkeit aufweisen. Um eine ungefähr gleiche Anzahl an zulässigen und unzulässigen Samples zu erhalten, soll eine von Kirscher (2021) entwickelte Modifikation, die sogenannte *Active-Learning-Strategy*, in die Informed Decomposition integriert werden. Daher sind einige Änderungen an den Abbildungen 1-2-a) und 1-2-b) erforderlich. Der neue Prozess ist in den Abbildungen 3-2 und 3-4 dargestellt.

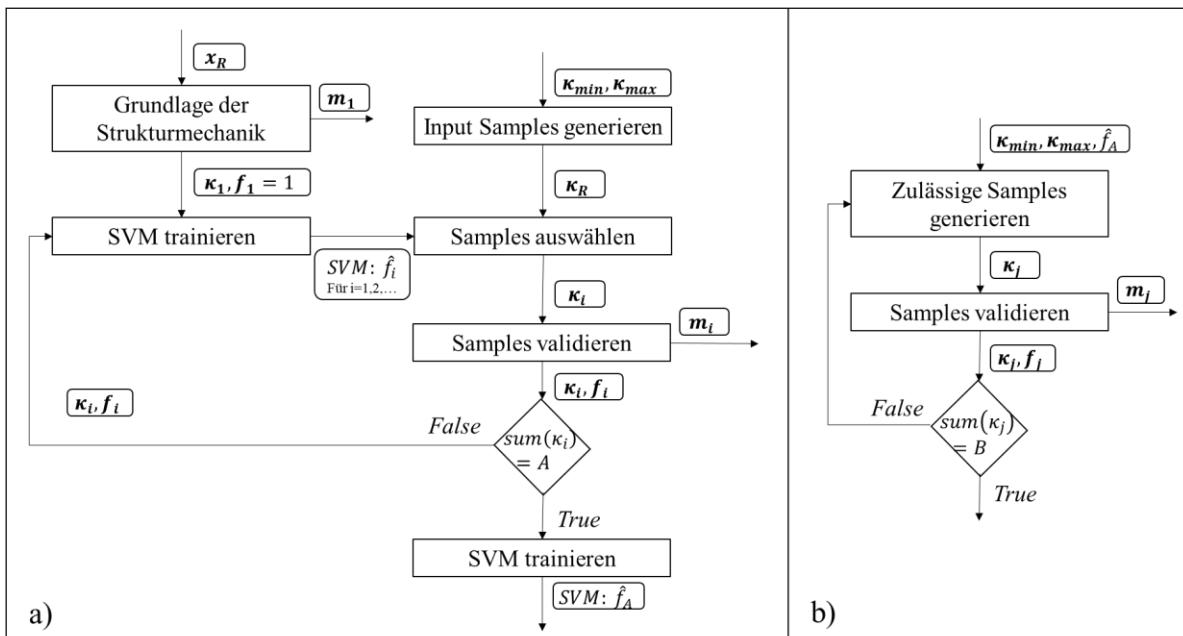


Abb. 3-2: Der Prozess zur Erzeugung von Samples basierend auf der Active-Learning-Strategy: in a) wird das erste SVM-Modell  $\hat{f}_1$  durch zufällig ausgewählte Designvariablen  $x_R$  trainiert. Mit  $\hat{f}_1$  als Ausgangspunkt wird in jeder Iteration neue Samples  $\kappa_R$  zulässig generiert und validiert. Dann durch alle vorhandenen Samples  $\text{sum}(\kappa_i)$  werden neu SVM  $\hat{f}_i$  trainiert. Wenn die Anzahl von allen vorhandenen Samples eine künstlich definierte Zahl  $A$  erreicht, wird das letzte SVM  $\hat{f}_A$  trainiert und wird der Prozess a) gestoppt. In b) werden zulässige Samples durch  $\hat{f}_A$  weiter generiert und validiert. Prozess b) wird beendet, wenn die Gesamtzahl der Massensamples eine künstlich definierte Zahl  $B$  erreicht.

Der erste Kerngedanke der Active-Learning-Strategy besteht darin, dass durch ein schichtweises Sampling das vollständig zufällige ersetzt wird. Jedem finiten Element in der Topologie wird eine Designvariable mit zufälliger Größe  $x_R$  zugewiesen, anschließend wird der entsprechende  $\kappa$ -Vektor errechnet. Da der  $\kappa$ -Vektor durch eine bestimmte Topologie erhalten wird, muss ein solcher  $\kappa$ -Vektor zulässig und mit  $f = 1$  markiert sein. Durch diesen

Prozess lassen sich zahlreiche zulässige  $\kappa$ -Vektoren bzw. Samples erhalten, die zusammen als  $\kappa_1$  bezeichnet werden. Alle  $\kappa_1$  sind zulässig, d. h.  $f_1=1$ . Die räumliche Verteilung von  $\kappa_1$  kann wie in Abbildung 3-3-a) dargestellt werden. Mit  $\kappa_1$  und  $f_1=1$  lässt sich die erste SVM  $\hat{f}_1$  trainieren.

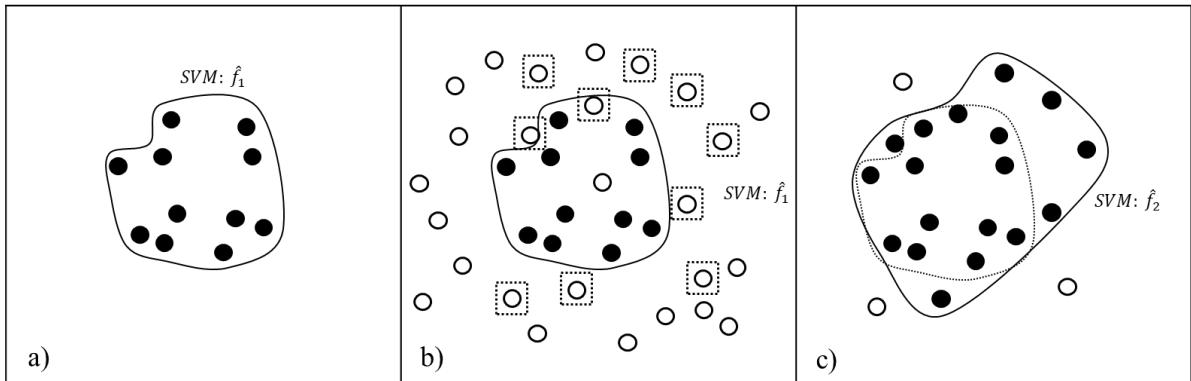


Abb. 3-3: Eine Iteration des schichtweisen Samplings. Abbildung a) zeigt die Verteilung von Samples  $\kappa_1$  und die dadurch trainierte SVM  $\hat{f}_1$ . In Abbildung b) wird ein vollständig zufälliges Sampling im Designraum durchgeführt, wobei die resultierenden Samples als hohle Kreise dargestellt werden. Die der Hyperebene von SVM  $\hat{f}_1$  am nächsten liegenden Samples  $\kappa_2$  werden ausgewählt (mit gestrichelten Linien umrahmt) und dann durch „Samples validieren“ beurteilt. Die zulässigen Samples darin werden durch die durchgehenden Kreise in Abbildung c) dargestellt und die neue SVM  $\hat{f}_2$  kann mit den vorhandenen Samples trainiert werden.

Wie in Abbildung 3-3-b) dargestellt, wird im nächsten Schritt, der Generierung von Input-Samples, eine große Anzahl von Samples im Designraum nach dem Zufallsprinzip generiert, aber daraus wird nur eine bestimmte Anzahl von Samples  $\kappa_2$  ausgewählt, die der Hyperebene von SVM  $\hat{f}_1$  am nächsten liegen. Diese Samples werden im Anschluss nacheinander validiert, um festzustellen, ob sie zulässig sind.

Der Verifizierungsprozess basiert auf der Strukturompimierung, deshalb soll MMA-Methode hierbei durchgeführt werden. Die in dieser Arbeit verwendete Gleichung des strukturellen Optimierungsproblems unterscheidet sich von Gleichung (2-25). Die Zielfunktion wird zur Masse  $m(x)$  und die Beschränkungen werden ebenfalls in eine sich auf den gegebenen  $\kappa$ -Vektor  $\kappa_{0,i}$  bezogene Beschränkung umgewandelt;  $\kappa_{0,i}$  ist ein gegebener Referenz-  $\kappa$ -Vektor. Das Ziel dieses Optimierungsproblems besteht darin, eine geeignete Designvariable  $x$  zu finden, die die Anforderung erfüllt, dass der Quotient aus der Differenz zwischen ihrem entsprechenden  $\kappa$ -Vektor  $\kappa(x)$  und dem gegebenen  $\kappa$ -Vektor  $\kappa_{0,i}$  geteilt durch  $\kappa_{0,i}$  kleiner als eine gegebene Toleranz  $\varepsilon$  ist, während ihre entsprechende Masse  $m(x)$  minimiert wird. Der Zweck eines  $\kappa_{0,i}$  im Nenner besteht darin, die Werte der vier Elemente in der Differenz  $|\kappa(x) - \kappa_{0,i}|$  auf einer einheitlichen Größenordnung zu halten. Hierbei werden folgende allgemeine Gleichungen aufgestellt:

$$\begin{aligned} & \min_x m(x) \\ & \text{subject to: } \frac{|\kappa(x) - \kappa_{0,i}|}{\kappa_{0,i}} \leq \varepsilon; \quad 0 < x_{min} < x \leq x_{max} \end{aligned} \quad (3-1)$$

$\kappa_{0,i}$  ist ein gegebener Referenz- $\kappa$ -Vektor, hierbei wird es durch den  $\kappa$ -Vektor in  $\kappa_2$  ersetzt, der aktuell verifiziert wird. Existieren geeignete Designvariablen, bedeutet dies, dass der aktuelle  $\kappa$ -Vektor zulässig ist, weshalb er nicht nur als zulässiges, sondern auch als Massensample gespeichert wird. Sofern keine geeignete Designvariable gefunden wird, ist der aktuelle  $\kappa$ -Vektor  $\kappa_{0,i}$  unzulässig. Auch dieser gespeichert, um die SVM zu trainieren. Die konkreten Konvergenzkriterien lauten hierbei wie folgt:

- Ist  $\Delta\kappa_i = \frac{|\kappa(w) - \kappa_{0,i}|}{\kappa_{0,i}}$  kleiner als eine manuell definierte Parameterbeschränkungstoleranz  $\varepsilon$ , bedeutet dies, dass das Sample zulässig ist, nämlich  $f_i = 1$ .
- Ist die manuell definierte maximale Iteration erreicht und  $\Delta\kappa$  größer als die Beschränkungstoleranz, wird das Sample als unzulässig markiert, nämlich  $f_i = -1$ .
- Ist die manuell definierte maximale Iteration erreicht und  $\Delta\kappa$  kleiner als die Beschränkungstoleranz, wird das Sample als zulässig markiert, nämlich  $f_i = 1$ .
- Ist die manuell definierte maximale Iteration noch nicht erreicht,  $\Delta\kappa$  größer als die Beschränkungstoleranz und die Veränderung der Designvariable kleiner als eine Schrittveränderungstoleranz  $1e - 5$ , wird das Sample als unzulässig markiert, nämlich  $f_i = -1$ .
- Ist die manuell definierte maximale Iteration noch nicht erreicht,  $\Delta\kappa_i$  kleiner als die Beschränkungstoleranz und die Veränderung der Designvariable kleiner als eine Schrittveränderungstoleranz  $1e - 5$ , wird das Sample als zulässig markiert, nämlich  $f_i = 1$ . Die obigen Bedingungen zeigen an, dass die MMA den optimalen Wert findet und dass in diesem Fall auch die optimale Masse  $m_i$  aufgezeichnet wird. Zudem wird dieses Sample verwendet, um das NN-Modell zu trainieren.

Nachdem alle Samples in  $\kappa_2$  überprüft wurden, wird der Vorgang wiederholt, wenn die Anzahl der Zulässigkeitssamples nicht die erforderliche Anzahl  $A$  erreicht. Diese Methode stellt sicher, dass die Samples jeder Iteration im Bereich des aktuellen zulässigen Designraums liegen, d. h. der in dieser Arbeit eingeführte Algorithmus nimmt den zulässigen Designraum als Ausgangsbereich, um den gesamten Designraum von Iteration zu Iteration zu erkunden.

Bei dem obigen Verfahren werden jedoch permanent neue Samples in der Nähe der Hyperebene der SVM und keines im inneren Bereich derselben, der dem zulässigen Bereich entspricht, erzeugt. Wenn die Anzahl der Samples in diesem Bereich nicht ausreicht, wird die Trainingsgenauigkeit des NN-Modells beeinträchtigt. Um dieses Problem zu lösen, wird ein weiterer Kerngedanke der Active-Learning-Strategie eingeführt, wie Abbildung 3-2-b) zeigt.

Durch den Prozess „Zulässige Samples generieren“ werden weiterhin Samples generiert und zunächst mit der im obigen Verfahren trainierten SVM  $\hat{f}_A$  validiert. Nur die von  $\hat{f}_A$  als zulässig markierten Samples werden gespeichert, um den zulässigen Bereich zu füllen. Allerdings können die Samples mittels SVM  $\hat{f}_A$  nicht absolut richtig klassifiziert werden. Deshalb ist es notwendig, die durch  $\hat{f}_A$  als zulässig markierten Samples durch das oben erwähnte MMA-Verfahren erneut zu validieren. Ziel ist, dass am Ende eine bestimmte Anzahl  $B$  an Masse-Samples existiert.

Der nächste Schritt besteht darin, das Metamodell durch die Zulässigkeits- und die Masse-

Samples zu trainieren, die mittels obiger Methode erhalten werden.

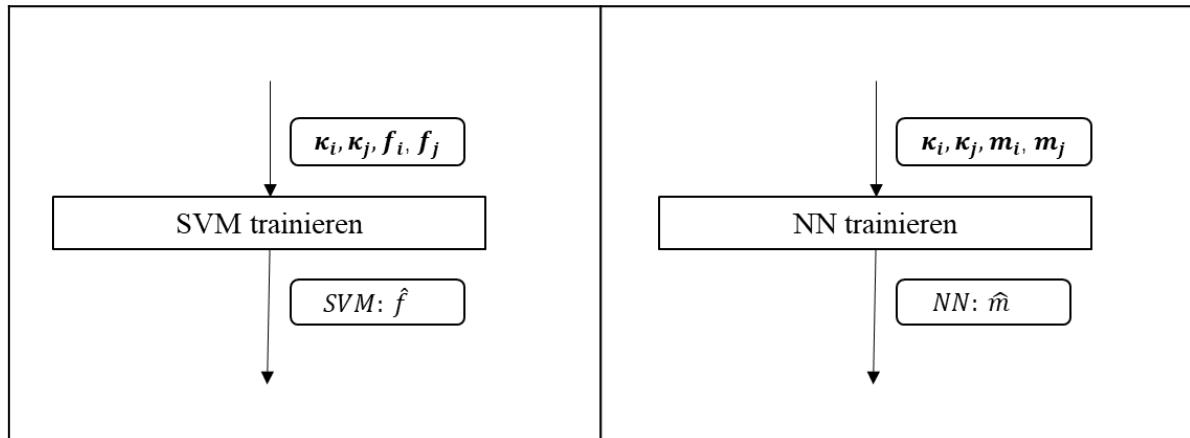


Abb. 3-4: Training des Metamodells anhand der in Abbildung 3-3 ermittelten Parameter

Wie bei der allgemeinen Datenverarbeitung im Machine-Learning, müssen die Samples zunächst gesplittet werden. Die Zulässigkeits- und die Masse-Sample werden in Trainings-, Test- und Validierungssatz aufgeteilt. In dieser Arbeit beträgt das Verhältnis von Trainings- und Testsatz beim Training der SVM [0.8:0.2], beim Training des NN beträgt das Verhältnis von Trainingssatz, Testsatz und Validierungssatz [0.8:0.1:0.1].

Hinsichtlich des SVM-Trainings ist Folgendes zu beachten: Da der Datensatz zufällig aufgeteilt wird, sind auch der verwendete Trainingssatz und das Trainingsergebnis, nämlich die Qualität der SVM, zufällig. Um eine SVM mit besserer Qualität erhalten werden kann, ist es notwendig, mehrere SVM gleichzeitig zu trainieren und das Modell mit der höchsten Genauigkeit (mindestens größer als 0.85) und einem falsch positiven Wert von weniger als 0.1 auszuwählen.

Der Grund, warum der falsch positive Wert begrenzt werden muss, besteht darin, dass die optimale Lösung unzulässig und somit nicht akzeptabel ist, wenn ein Teil des unzulässigen Bereichs im Designraum von der SVM als zulässig erachtet wird und die optimale Lösung bei der nachfolgenden Optimierung in diesen Bereich fällt. Wenn hingegen ein Teil des zulässigen Bereichs im Designraum von der SVM als unzulässig erachtet wird, resultiert daraus eine zwar suboptimale, aber zulässige Lösung.

Die Obergrenze der in dieser Arbeit festgelegten Anzahl zu trainierender SVM beträgt 60 und die Größe des Parameters  $c$  in Gleichung 2-16 wird nach dem Training je 15 SVM angepasst. Sobald eine trainierte SVM die Anforderungen erfüllt, wird sie gespeichert. Sind bereits vor Abschluss des Trainings aller 60 SVM acht gespeicherte SVM vorhanden, wird der Trainingsvorgang unterbrochen, die gespeicherte SVM mit der höchsten Genauigkeit ausgewählt und diese für die Durchführung der anschließenden Optimierungsaufgaben verwendet.

Die Zufälligkeit von NN sollte auch beim Training derselben berücksichtigt werden. Wie bei SVM, werden in dieser Arbeit auch mehrere NNs trainiert. Diese besitzen unterschiedliche Strukturen, die zwischen einem und fünf Hidden Layer und ein bis 19 Neuronen pro Layer aufweisen. Auch hier wird das NN mit der besten Genauigkeit ausgewählt.

Die zentralen Parameter beim Training der Metamodelle sind in folgender Tabelle zusammengefasst:

	Satzeinstellung	Acc	fn	c	Anzahl
SVM	[0.8:0.2]	>0.85	<0.1	[1,2,3,4]	60
	Satzeinstellung	Hidden Layer		Neuron	
NN	[0.8:0.1:0.1]	[1,2,3,4,5]		[1,3,5...15,17,19]	

Tabelle 3-2: Die zentralen Parameter beim Training der Metamodelle. Acc: Genauigkeit; fn: falsch positiver Wert; Anzahl: Menge der zu trainierenden SVM

### 3.3 Systemoptimierung

Als nächstes wird das System mit dem ausgewählten Metamodell optimiert, wie Abbildung 1-2-c), um die Referenz- $\kappa$ -Vektoren für beide Komponenten zu erhalten. Das entsprechende Optimierungsproblem lautet wie folgt:

$$\begin{aligned}
 & \min_{\kappa} \hat{m}(\kappa) \\
 & \text{subject to: } \hat{f}(\kappa) = 1; \\
 & \quad u(\kappa) \leq u_c + \varepsilon_u; \\
 & \quad x^T \cdot \mathbf{K}(\kappa) \cdot x \geq 0 \quad \forall x \in R; \\
 & \quad \kappa_{min} \leq \kappa \leq \kappa_{max}
 \end{aligned} \tag{3 - 2}$$

Die erste Einschränkung erfordert, dass die Designvariable  $\kappa$  durch die SVM als zulässig verifiziert werden muss. Der Parameter  $u_c$  in der zweiten Einschränkung ist die Verschiebungsbedingung des freien Endes des Modells, für die in dieser Arbeit ein Wert von 1 mm definiert wird. Die dritte Einschränkung soll anhand des zweiten Merkmals der Steifigkeitsmatrix, nämlich positiv- oder semidefinite quadratische Form, erfüllt werden. Die Zielfunktion  $\hat{m}(\kappa)$  wird durch NN berechnet.

Da die Parameter  $\hat{m}(\kappa)$  und  $\hat{f}(\kappa)$  Metamodelle sind und nicht abgeleitet werden können, ist das MMA-Verfahren für die Optimierung von Gleichung (3-2) nicht nutzbar. In der vorliegenden Arbeit wird die *Particle-Swarm-Optimization* (PSO) verwendet, um dieses Problem zu lösen. Die entsprechende PSO-Gleichung lautet wie folgt:

$$P(\kappa) = \hat{m}(\kappa) + \alpha \cdot \left( \frac{\max(0, u(\kappa) - u_c - \varepsilon_u)}{u_c} + (\hat{f}(\kappa) - 1) + \min(0, x^T \cdot \mathbf{K}(\kappa) \cdot x) \right) \tag{3 - 3}$$

Hierbei ist  $\alpha$  ein künstlich definierter Straffaktor, der so groß sein sollte, dass der Wert in Klammern, falls nicht null, eine höhere Größenordnung als  $\hat{m}(\kappa)$  aufweist. In dieser Arbeit wird  $\alpha$  der Wert 2000 zugewiesen. Nach der Lösung der PSO können die Referenz- $\kappa$ -Vektoren jeder Komponente  $\kappa_{0,K1}$  und  $\kappa_{0,K2}$  erhalten werden. Um die bestmöglichen Ergebnisse zu erzielen, wird die PSO mehrmals ausgeführt, um mehrere Ergebnissätze zu generieren, von denen die Ergebnisse  $\kappa_0$  mit der niedrigsten Gesamtmasse  $\hat{m}(\kappa)$  für die Komponentenoptimierung ausgewählt werden.

### 3.4 Komponentenoptimierung

Das Optimierungsproblem der Komponentenoptimierung ist gleiche wie die Gleichung (3-1) und wird auch durch MMA gelöst. Der Zweck besteht darin, die Designvariable  $x$ , die entsprechende spezifische Struktur und Masse anhand  $\kappa_{0,K1}$  und  $\kappa_{0,K2}$  zu finden, und die sind die Endergebnisse vom Verfahren *Informed Decomposition*.

Um die Genauigkeit der Informed Decomposition zu bewerten, müssen abschließend die endgültigen Ergebnisse überprüft werden. Die hierfür angewandte Methode besteht darin, ein monolithisches Optimierungsproblem für die gleiche Topologie aufzubauen und MMA zu verwenden um das zu lösen. Dieses monolithische Optimierungsproblem wird wie folgt ausgedrückt:

$$\begin{aligned}
 & \min_x m(x) \\
 & \text{subject to: } \frac{u(x)}{u_C} = 1; \\
 & \quad K \cdot U = F; \\
 & \quad x_{min} \leq x \leq x_{max}
 \end{aligned} \tag{3 - 4}$$

## 4 1-D Balken

Die direkte Programmierung dreidimensionaler Strukturen erfordert die Verarbeitung großer Datenmengen, und die Programmlaufzeit kann bis zu mehreren Tagen dauern. Um den Rechenaufwand zu reduzieren, beginnt diese Arbeit mit eindimensionalen Strukturoptimierungsproblemen, passt die Programmstruktur an, sammelt Erfahrungen mit der Parameterabstimmung und verallgemeinert sie für die spätere Optimierung von höherdimensionalen Strukturen.

Die Methodik zur Behandlung von ein-, zwei- und dreidimensionalen Strukturen ist grundsätzlich gleich und wurde bereits in Kapitel 3 vorgestellt. In diesem Kapitel und in den folgenden Kapiteln werden nur die darin geänderten Teile vorgestellt.

### 4.1 Problemdefinition

Die in diesem Abschnitt verwendete Struktur besteht aus zwei verbundenen I-Trägern. Die Schnittstellenparameter des I-Trägers sind in Abbildung 4-1 dargestellt. Die äußere Kraft beträgt 49,03 kN und wirkt auf das rechte Ende des rechten Bauteils, siehe Abbildung 3-1 Freiheitsgrad 5.

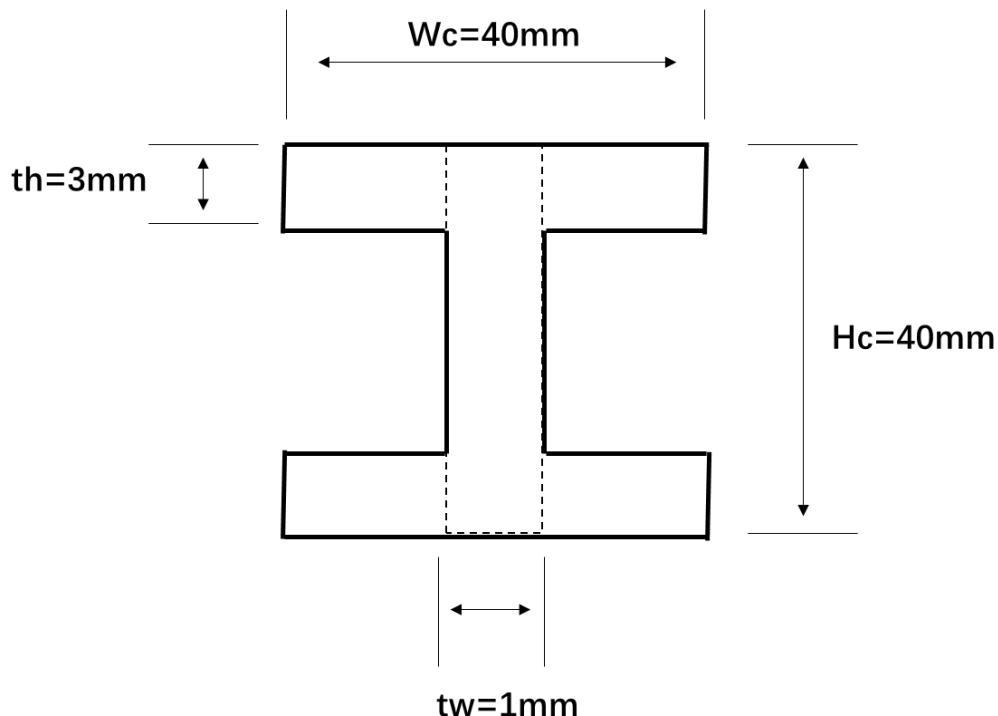


Abb. 4-1: Der Teil mit durchgehender Linie ist die größte Struktur, der Teil mit gestrichelter Linie ist die kleinste Struktur und die Länge jedes I-Trägers beträgt 300 mm.

Die Designvariable ist hierbei die Breite des Seitenquerschnitts  $w$ .

$$x = W, \quad t_w \leq W \leq W_c \quad (4 - 1)$$

Jede Komponente wird durch die Finite-Elemente-Analyse in 10 Finite-Elemente in Längsrichtung unterteilt, d. h.  $n_{ele} = 10$ . Und die eindimensionale Steifigkeitsmatrix ist wie die Gleichung (4-2) gezeigt.

$$K = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}, \quad I = W \cdot H_c^3 / 12 \quad (4-2)$$

Im Prozess Active Learning Strategy wird vor dem Training jedes SVMs 800 neue  $\kappa$ -Vektoren bzw. Samples generiert, d. h. In  $\kappa_1$  sowie  $\kappa_i$  werden je 800 Samples enthalten. Als ein Beispiel stellt die Abbildung (4-2) die räumliche Verteilung von  $\kappa_1$  dar.

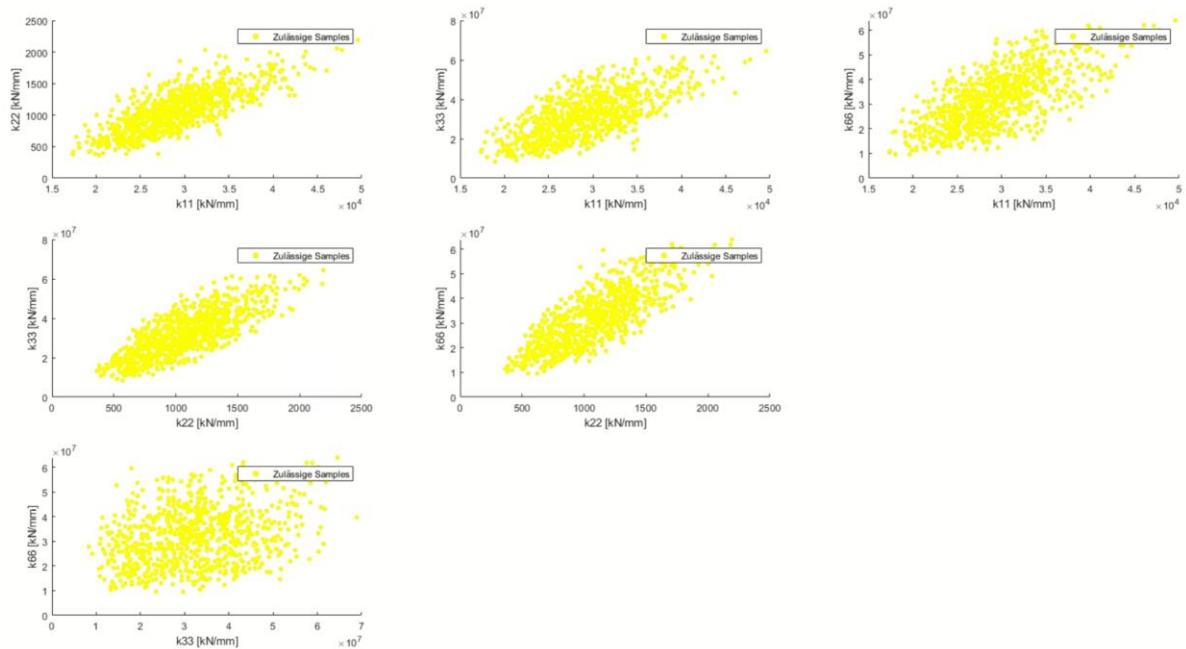


Abb. 4-2: Die Verteilung des ersten 800 Samples. Zulässiges Sample wird als gelb bemerkt und unzulässiges Sample wird als blau bemerkt. Der gelbe Bereich kann als zulässiger Designraum bezeichnet werden. Weil es im  $\kappa$ -Vektor 4 Elemente ( $\kappa_{11}, \kappa_{22}, \kappa_{33}, \kappa_{66}$ ) gibt, wird hierbei das Verhältnis zwischen je zwei Parameter durch ein Bild beschrieben.

Beim „Samples validieren“ sowie Komponentenoptimierung ist die Toleranz  $\varepsilon$  in der Gleichung (3-1) in diesem Kapitel als  $1e - 3$  definiert.

Darüber hinaus ist zu beachten, dass die Gesamtmasse wie die Gleichung (4-3) normalisiert

werden muss, damit deren Wert in einer gewissen Größenordnung beschränkt werden.

$$m = \frac{100}{A_{max} \cdot n_{ele}} \cdot \sum_{e=1}^{n_{ele}} w_e \cdot h \quad (4-3)$$

Auf diese Weise kann die Gesamtmasse  $m$  auf das Intervall [1 – 100] begrenzt werden, was die Verständlichkeit erleichtert. Darin beschreibt  $w_e$  die Breite von einer einzelnen Finite Elemente.

Außerdem erfordert MMA auch die Ableitungsfunktion der Parameter  $m(w)$  und  $K(w)$ , und ihr Ausdruck ist wie folgt (O'Connell et al., 1976, Abschnitt 2.1.2):

$$\frac{\partial m(w)}{\partial w_e} = \frac{100 \cdot h}{A_{max} \cdot n_{ele}} \quad (4-4)$$

$$\frac{\partial K(w)}{\partial w_e} = T_G^T \cdot \frac{\partial K_{Kombiniert}(w)}{\partial w_e} \cdot T_G \quad (4-5)$$

Während des ganzen Prozesses von der Erzeugung von Samples wird in diesem Kapitel 8800 Zulässigkeit-Samples für das Training von SVM und 9000 Masse-Samples für das Training von NN, nämlich  $A = 8800$  und  $B = 9000$  in Abbildung 3-2, generiert.

Alle in diesem Abschnitt aufgeführten wichtigen Parameter sind in der folgenden Tabelle zusammengefasst:

Topologie			Finite Elemente	$\kappa_1$	$\kappa_i$	A	B
Länge	Breite	Höhe	$n_{ele} = 10$	800 Samples	800 Samples	8800 Samples	9000 Samples
300mm	1~40mm	40mm					

Tabelle 4-1: Wichtige Parameter für die Optimierung von 1-D Balken

## 4.2 Ergebnisse

### 4.2.1 Parameterstudie

Um geeignete Ergebnisse zu erhalten, müssen einige Schlüsselpараметer im Algorithmus angepasst werden:

#### Ratio

Da die erste SVM von Aktive Learning Strategy als Ausgangspunkt für die Erkundung des gesamten Designraums dient, muss geprüft werden, ob die räumliche Verteilung der ersten 800 Samples, die für das Training der ersten SVM verwendet werden, einen Einfluss auf das

Endergebnis hat. Um die Verteilung der ersten 800 Samples zu kontrollieren, wird hier eine Kombinationsformel aus vollständig zufälligem Sampling und nicht zufälligem Sampling eingeführt. *Ratio* wird als Parameter verwendet, um die Teilnahmequote der beiden Samplingsverfahren anzupassen. Diese Kombinationsformel lässt sich grob wie folgt ausdrücken:

$$x = \alpha \cdot Z + (1 - \alpha) \cdot NZ \quad \text{mit } \alpha = [0,1] \quad (4 - 6)$$

$x$  ist die Designvariable,  $\alpha$  beschreibt den Parameter Ratio.  $Z$  ist für das vollständig zufällige Sampling und  $NZ$  ist für das nicht zufällige Sampling. Es handelt sich bei  $\alpha = 0$  um ein nicht zufälliges Sampling, im Gegenteil, wenn  $\alpha = 1$ , handelt es sich um ein vollständig zufälliges Sampling.

Als nächstes werden die Verteilungsergebnisse in den drei Fällen von  $\alpha = 1$ ,  $\alpha = 0.5$  und  $\alpha = 0.1$  besprochen:

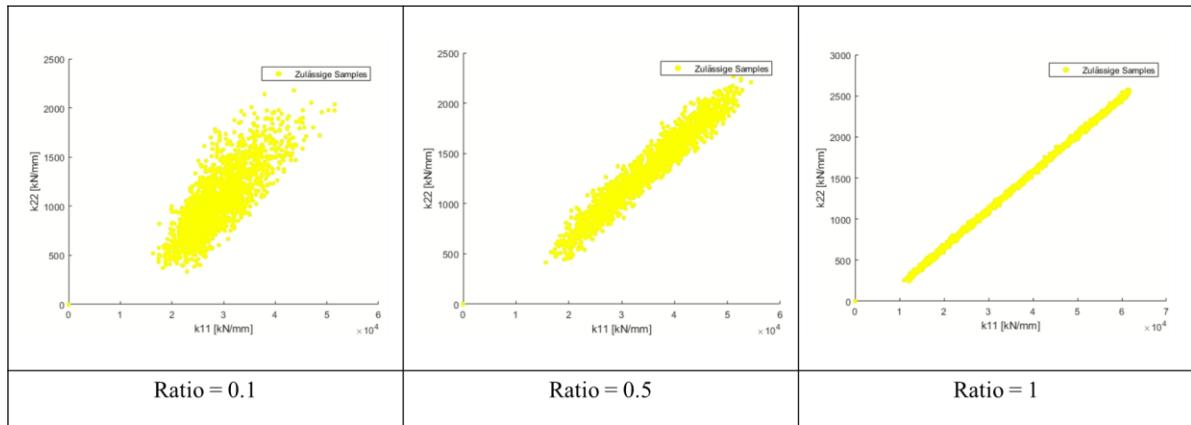


Abb. 4-3: Die Verteilung von  $\kappa_{11}$  und  $\kappa_{22}$  des ersten 800 Samples mit Ratio=0.1, 0.5 und 1.

Aus den drei obigen Abbildungen ist ersichtlich, dass sich die Samples bei  $Ratio = 1$  hauptsächlich in der Mitte des Designraums konzentrieren. Wenn das Ratio abnimmt, wird der aus Samples zusammengesetzte zulässige Raum schmäler und länger und beginnt sich in der Richtung der Grenze des Designraums auszudehnen. Die endgültigen Sampling-Ergebnisse, die mit den drei oben genannten Ausgangszuständen erzielt werden, weisen ebenfalls entsprechende Merkmale auf:

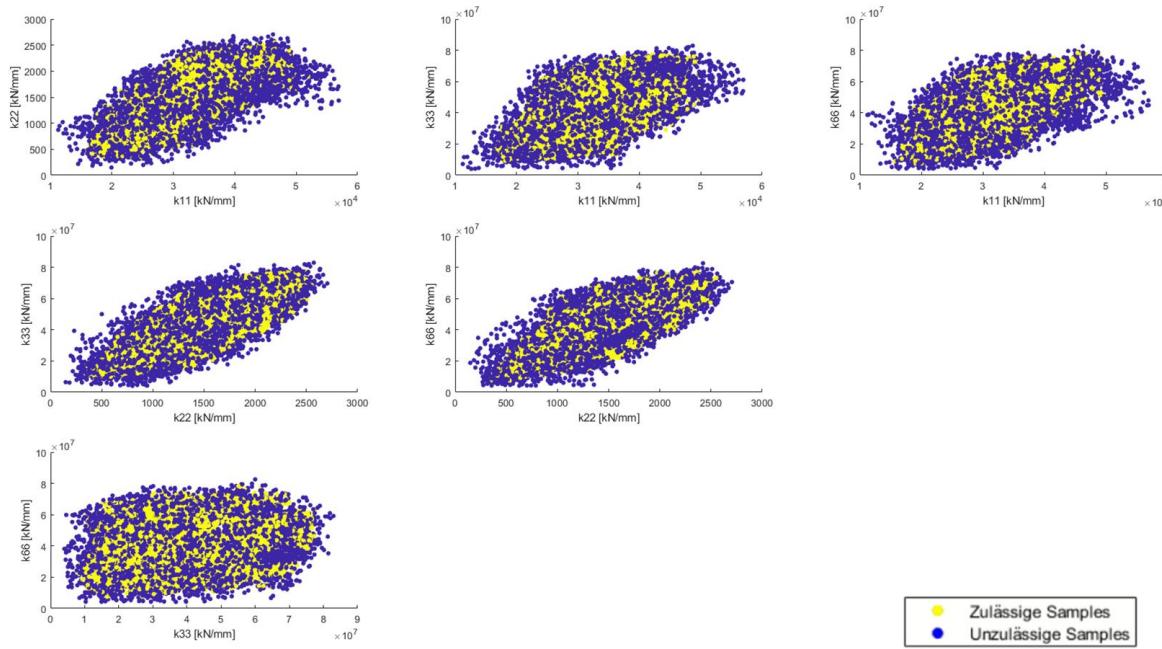


Abb. 4-4: Die Verteilung des gesamten 8800 Zulässigkeit-Samples mit  $\text{Ratio}=1$

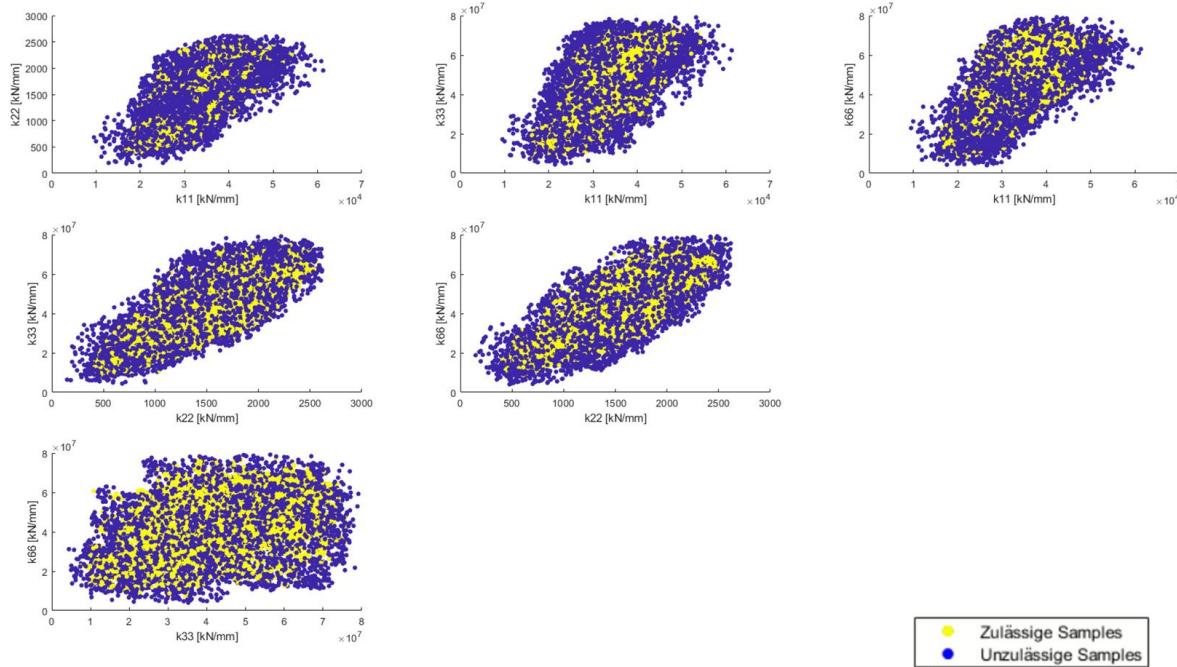


Abb. 4-5: Die Verteilung des gesamten 8800 Zulässigkeit-Sample mit  $\text{Ratio}=0.5$

Beim Vergleich der Teilabbildungen an der Position (1, 1) in diesen drei Figuren 4-4, 4-5 und 4-6 ist es zu erkennen, dass der Designraum im Fall  $\text{Ratio} = 1$  in Richtung von links unten nach rechts oben ungenügend exploriert ist, während der Designraum im Fall  $\text{Ratio} = 0.1$  in

Richtung von links oben nach rechts unten ungenügend exploriert ist. Der Fall  $Ratio = 0.5$  liegt irgendwo dazwischen.

In diesem Kapitel wird  $Ratio = 1$  gewählt, weil in der Mitte des Designraums im Fall  $Ratio = 1$  es mehr Samples gibt, was zum Training der Metamodelle förderlich ist.

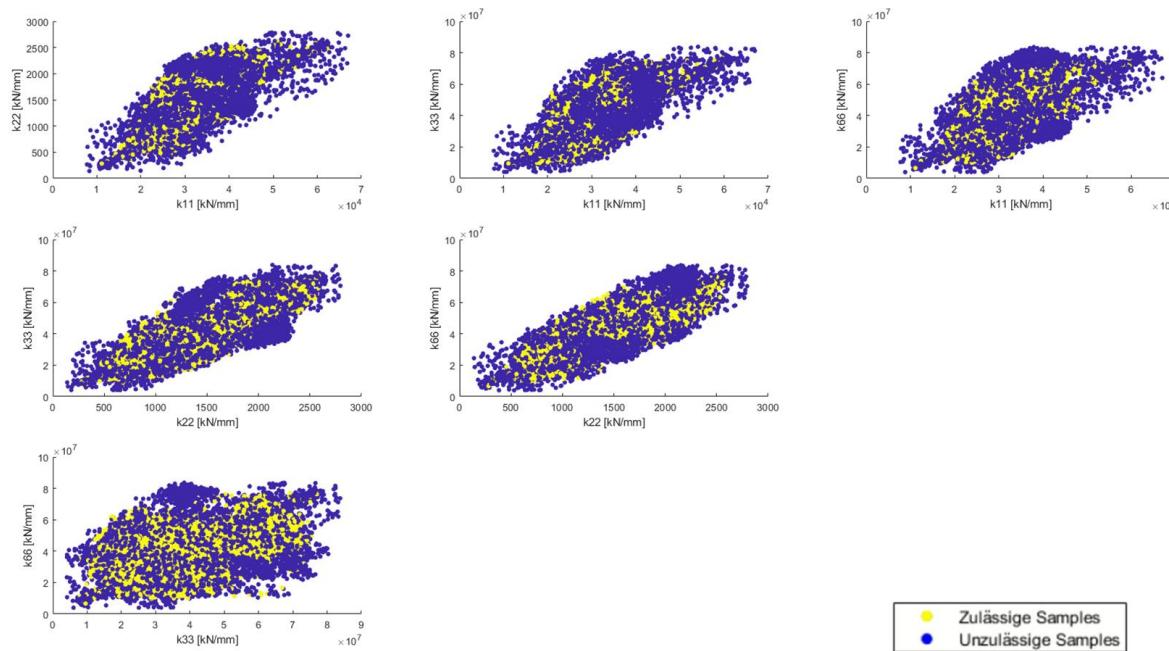


Abb. 4-6: Die Verteilung des gesamten 8800 Zulässigkeit-Sample mit  $Ratio=0.1$

## Fraction

Als nächstes wird der Effekt eines anderen Parameters *Fraction* betrachtet. *Fraction* bezieht sich auf die Anzahl der Iterationen im Sampling-Prozess in Abbildung 3-2-a). Beispielweise für 8800 Zulässigkeits-Samples bedeutet *Fraction* = 5, wenn insgesamt 5 SVMs während des Sampling-Prozesses trainiert werden und in jeder Iteration 1600 Samples generiert werden (Das erste SVM und die entsprechenden 800 Samples werden hierbei nicht gezählt). Im Folgenden werden hauptsächlich drei Fälle mit *Fraction* = 5, 10, 20 analysiert.

Aus den folgenden drei Abbildungen ist ersichtlich, dass der Explorationsgrad zum Designraum von den Fällen mit verschiedenen Fractions grundsätzlich gleich ist. Allerdings sind die Verteilungen von zulässig und unzulässig Samples im Fall *Fraction* = 10 und 20 in Vergleich zu der im Fall *Fraction* = 5 gleichmäßiger. Deshalb ist *Fraction* = 10 unter Berücksichtigung des Zeitaufwands geeigneter.

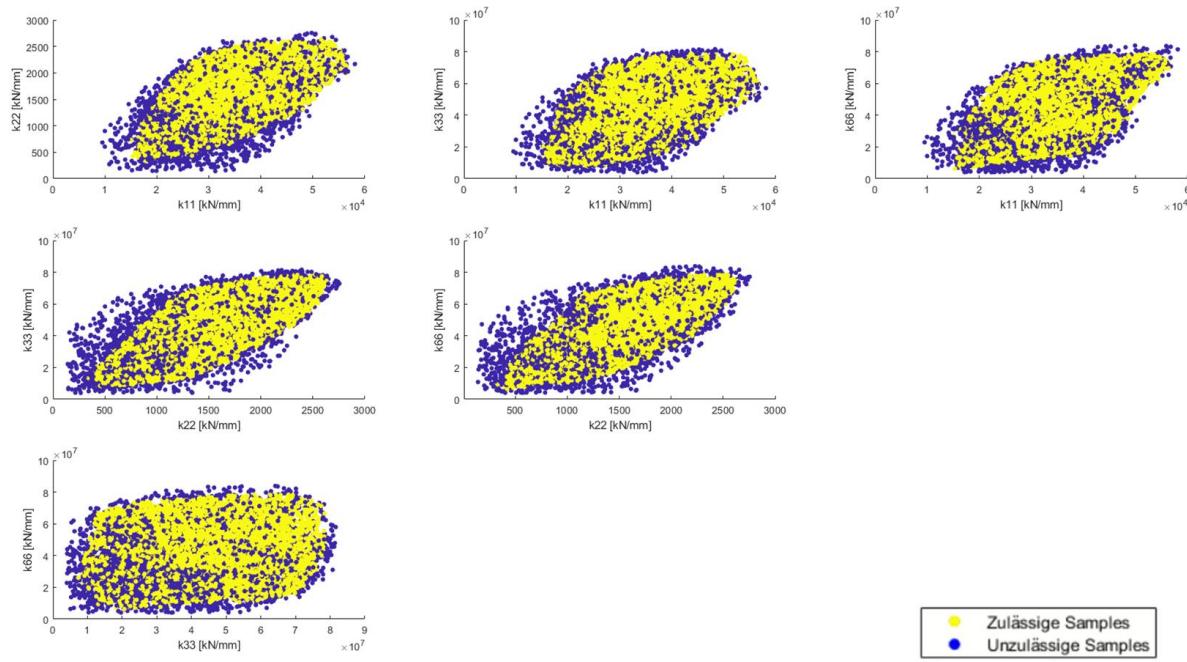


Abb. 4-7: Die Verteilung des gesamten 8800 Zulässigkeit-Samples mit Fraction=5

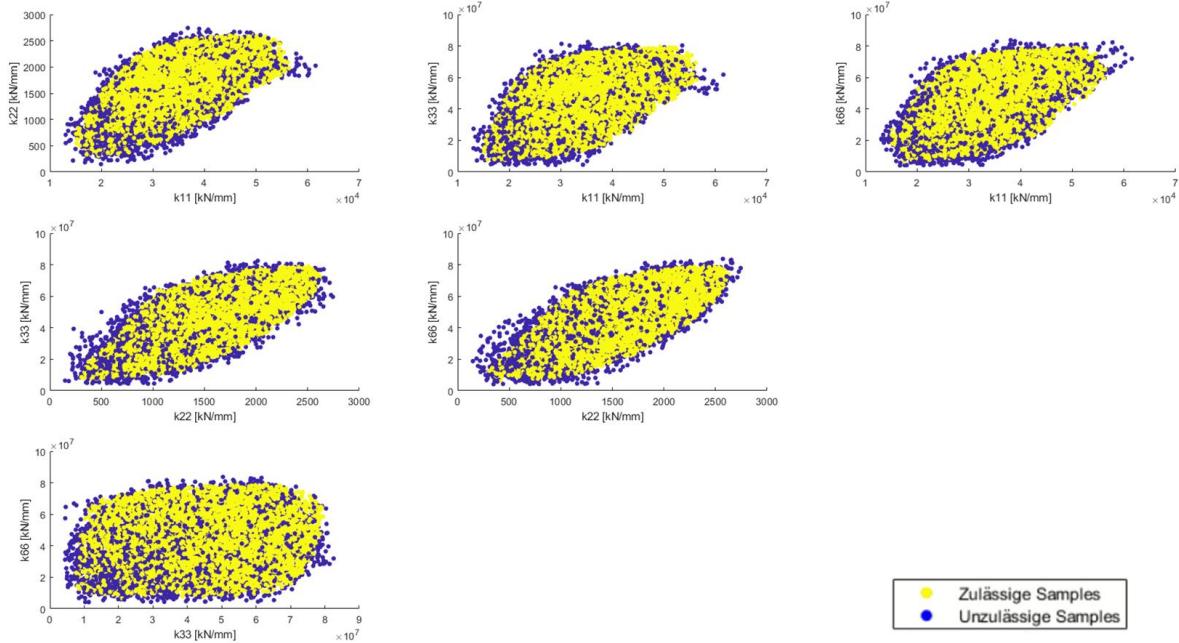


Abb. 4-8: Die Verteilung des gesamten 8800 Zulässigkeit-Samples mit Fraction=10

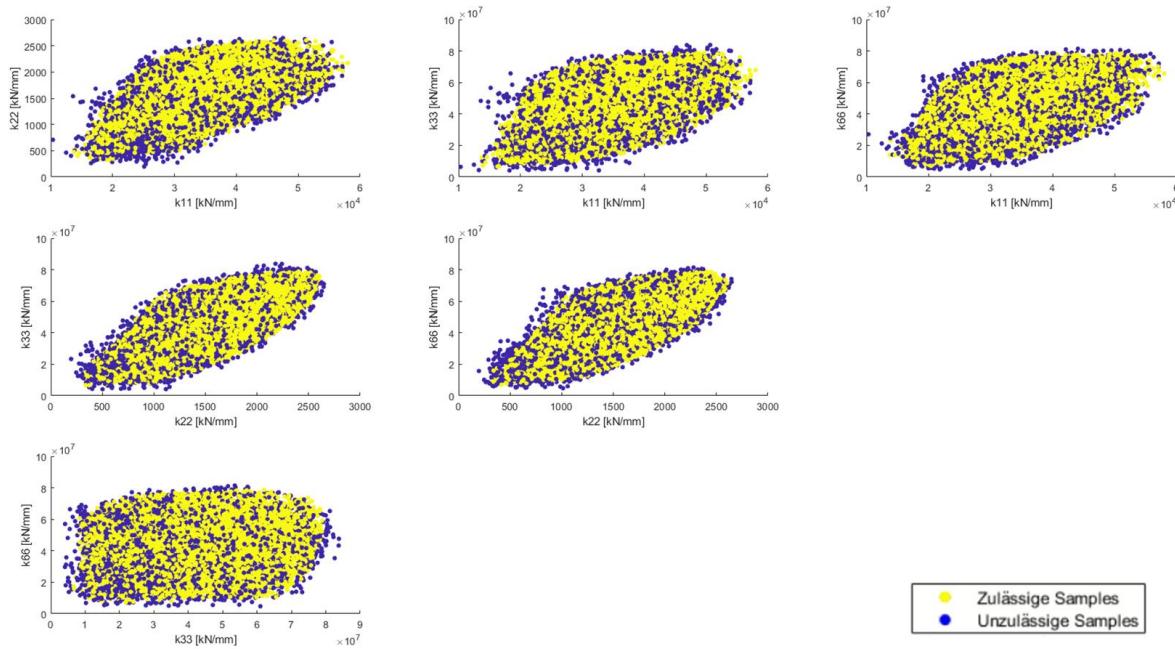


Abb. 4-9: Die Verteilung des gesamten 8800 Zulässigkeit-Samples mit Fraction=20

## Andere Parameter

Das erste, was zu beachten ist, ist der Parameter  $\varepsilon$  in der Gleichung (3-1). Dieser Parameter repräsentiert die Fehlertoleranz zwischen dem optimierten  $\kappa$ -Vektor und dem Referenz- $\kappa$ -Vektor. Es ist zu beachten, dass einige Teile des zulässigen Bereiches im Designraum sehr dünn sein können, wie ein Blatt im dreidimensionalen Raum. Ein zu groß  $\varepsilon$  kann dicker sein als der zulässige Bereich, siehe Abbildung unten, was die Einschränkung in der Gleichung (3-1) bedeutungslos macht. Um dieses Problem zu vermeiden, wird hier ein sehr kleiner Wert  $\varepsilon = 1e - 3$  gewählt.

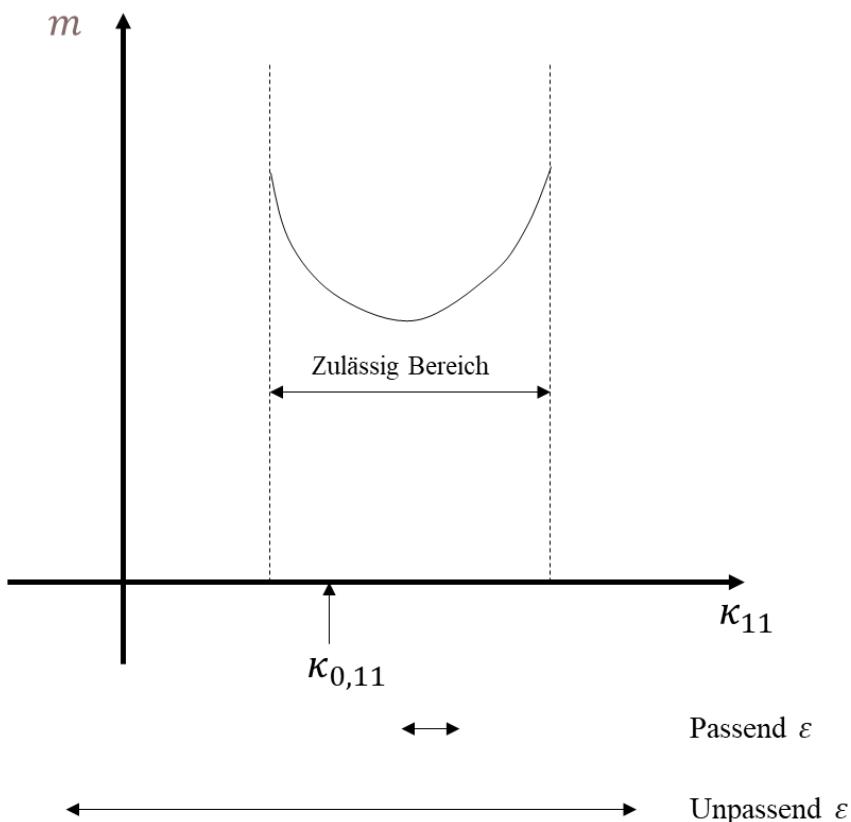


Abb. 4-10: Die Beziehung zwischen  $\kappa_{11}$  an einer gewissen Position im Designraum und der Masse  $m$ . Diese Beziehung wird durch eine durchgezogene Kurve dargestellt. Der Bereich zwischen den gestrichelten Linien stellt den zulässigen Bereich dar. Gemäß der Einschränkung in Gleichung (3-1) sollte  $\kappa(w)$  nahe  $\kappa_0$  liegen. Wenn der Wert von  $\varepsilon$  zu groß ist, dann fällt  $\kappa(w)$  mit hoher Wahrscheinlichkeit im unzulässigen Bereich, was die Einschränkung bedeutungslos macht.

Danach soll die Anzahl von zulässigen und unzulässigen Samples so gleich wie möglich sein. Dies kann erreicht werden, indem bei ‚Samples auswählen‘ in Abbildung 3-2-a) ein Filter hinzugefügt wird. Anstatt einfach 800 Samples auszuwählen, die der Hyperebene am nächsten sind, werden hierbei 400 Samples an beiden Seiten der Hyperebene ausgewählt, die der Hyperebene am nächsten liegen. Da SVM immer Klassifikationsfehler hat, wird das Verhältnis der Anzahl von zulässig und unzulässig Samples am Ende nicht absolutes 1:1 sein, aber der Unterschied zwischen beiden Anzahlen wird stark reduziert.

Zum Schluss gibt es noch ein Problem zu beachten: das vorhandene SVM ist nicht genau genug, sodass eine Lücke zwischen dem gemäß SVM definierten zulässigen Bereich und dem tatsächlich zulässigen Bereich besteht, wie in Abbildung 4-11 dargestellt. Die Samples innerhalb dieser Lücke sind eigentlich nicht zulässig und haben somit nicht die korrekte Masse, werden aber trotzdem als Massensample nach dem SVM ausgewählt. Dies kann zu einem sehr steilen, aber falschen Abstieg des NN-Modells innerhalb dieser Lücke führen. Da die Zielfunktion der Systemoptimierung darin besteht, ein Design mit der geringsten Masse zu finden, kann wegen dieses Problems es sehr leicht passieren, dass die Ergebnisse der Systemoptimierung in diese Lücke fallen und unzulässig sind.

Die Lösung besteht darin, die Samples in dieser Lücke herauszufinden und die Masse von solchen Samples mit einem Koeffizienten zu multiplizieren, so dass es in der Lücke ein steigender Verlauf gibt. Der Koeffizient wird in diesem Kapitel als 1.2 ausgewählt. Solche unzulässigen Samples in der Lücke werden als *negative Massensamples* genannt.

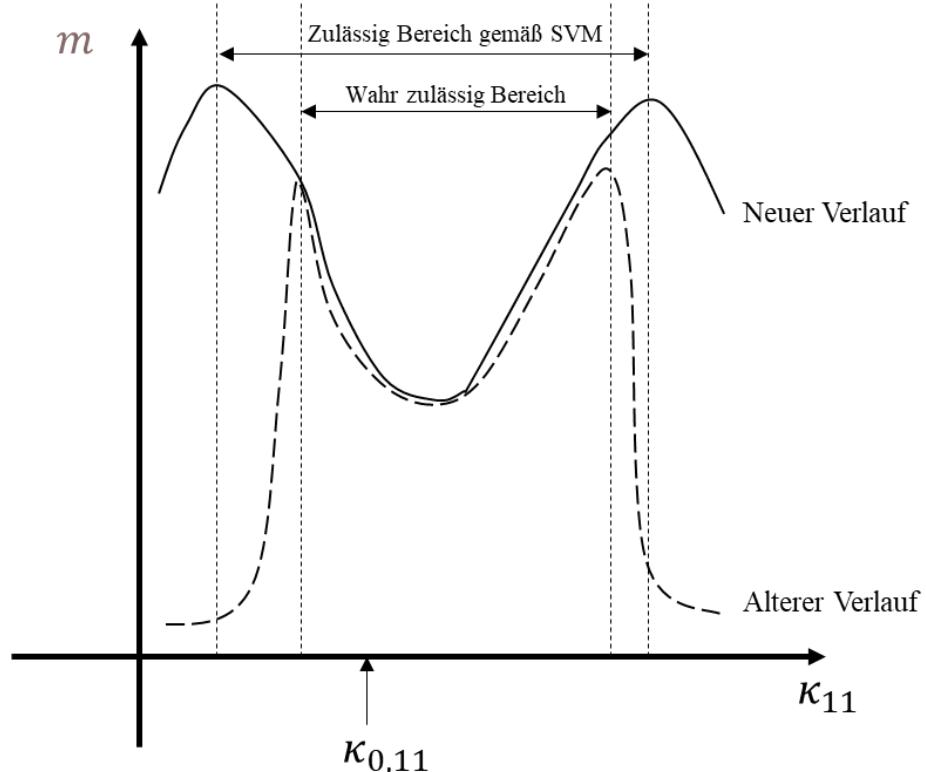


Abb. 4-11: Der Verlauf vor und nach der Einführung von negativen Massensamples. Zwischen dem gemäß SVM definierten zulässigen Bereich und dem wahr zulässigen Bereich gibt es eine Lücke und die Masse von allen Samples in dieser Lücke werden mit einem Koeffizienten multipliziert werden.

#### 4.2.2 Endergebnisse

Nun können die finalen Ergebnisse der eindimensionale Strukturoptimierung analysiert werden. Die Werte von ausgewählten Hauptparametern sind wie Tabelle 4-2 gezeigt.

Die Abbildungen 4-12 und 4-13 stellen die Verteilung der Zulässigkeitssamples und der Massensamples dar. In der Abbildung 4-13 ist es ersichtlich, dass die Grenze des zulässigen Bereiches noch nicht glatt ist, was darauf hindeutet, dass der gesamte Designraum noch nicht vollständig erkundet ist. Aber gibt es bis jetzt schon genügend Samples in der Mitte des zulässigen Bereichs, sodass für die meisten Designs die mit solchen Samples trainierten Metamodelle ausreichen.

Samples Anzahl	$\varepsilon$	Fraction	Negative Massensample	Ratio
8800	1e-3	10	*1.2	1

Tabelle 4-2: Die Werte von den Hauptparametern

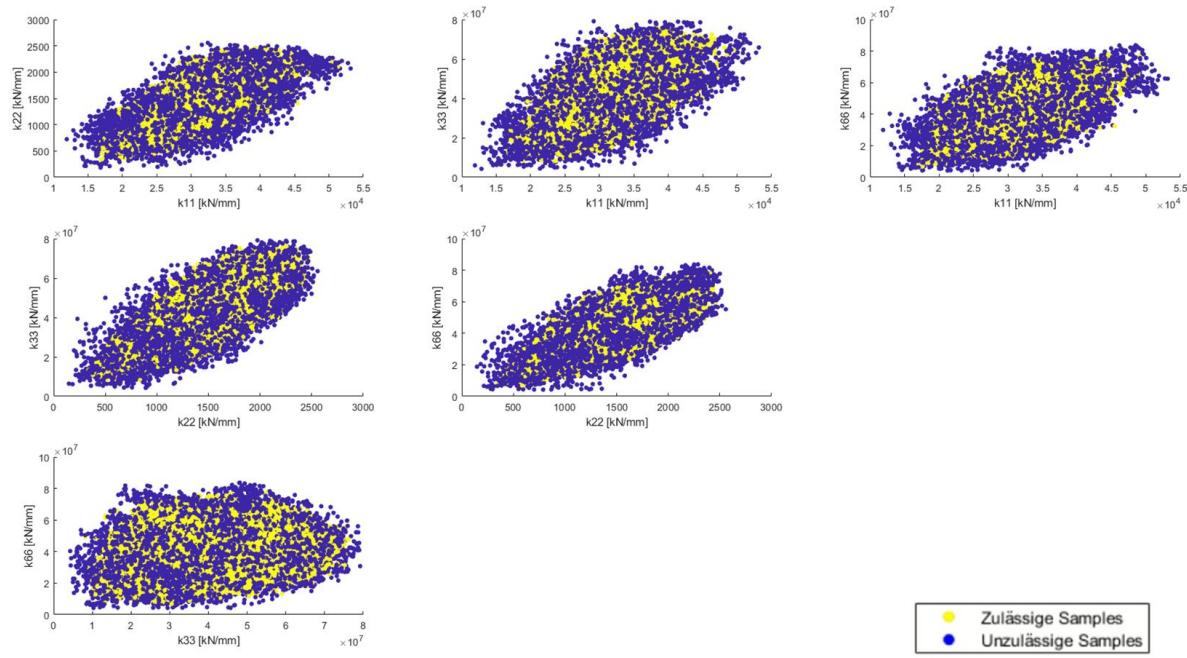


Abb. 4-12: Die Verteilung des gesamten 8800 Zulässigkeit-Samples

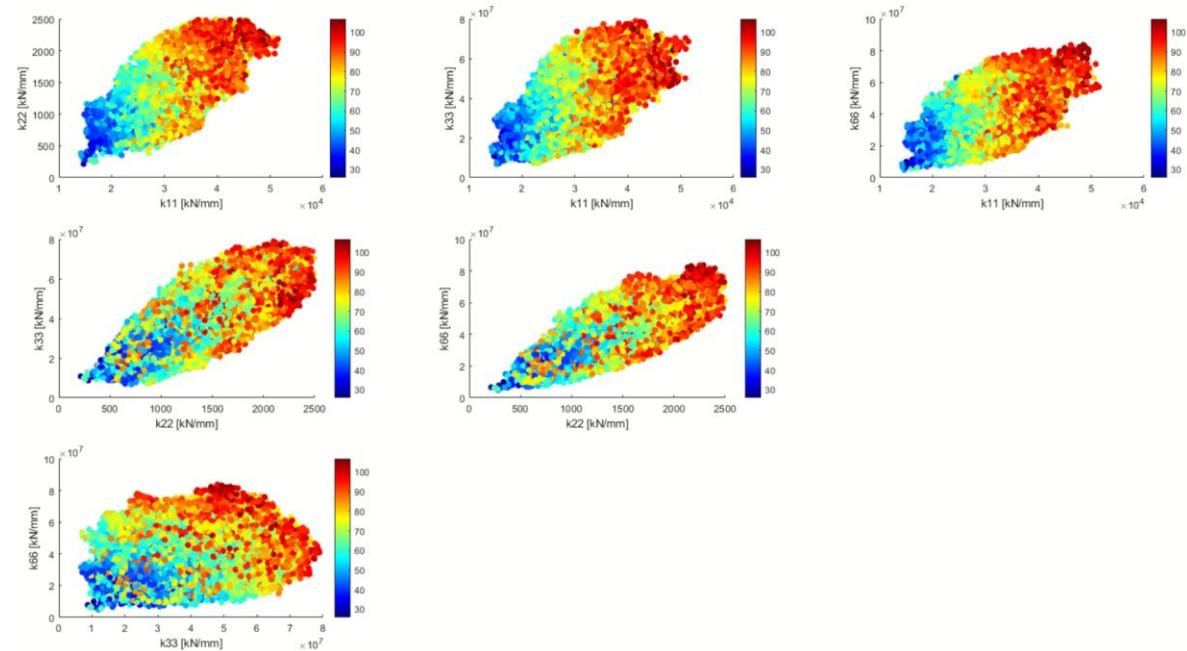


Abb. 4-13: Die Verteilung des gesamten 8800 Masse-Samples (darin keine negativen Massensamples enthält)

	Erste Komponente		Zweite Komponente	
	$\kappa_{1,optimal}$	$m_{1,optimal}$	$\kappa_{2,optimal}$	$m_{2,optimal}$

Informed Decomposition	$\begin{bmatrix} 4.0203e + 04 \\ 1.5232e + 03 \\ 5.5386e + 07 \\ 3.7762e + 07 \end{bmatrix}$	65.1551	$\begin{bmatrix} 2.1324e + 04 \\ 749.9938 \\ 3.0767e + 07 \\ 1.4961e + 07 \end{bmatrix}$	36.6195
Monolithische Optimierung	$\begin{bmatrix} 4.2940e + 04 \\ 1.7091e + 03 \\ 6.0255e + 07 \\ 4.3030e + 07 \end{bmatrix}$	69.0850	$\begin{bmatrix} 1.6024e + 04 \\ 477.8864 \\ 2.3089e + 07 \\ 7.2604e + 06 \end{bmatrix}$	29.3286

Tabelle 4-3: Die optimalen Parameter von zwei Komponenten nach monolithischer Optimierung sowie nach Informed Decomposition

Tabelle 4-3 listet die Optimierungsergebnisse von monolithischer Optimierung und von Informed Decomposition auf. Die optimale Gesamtmasse ( $m_1 + m_2$ ) durch das informed Decomposition ist etwas größer als die durch die monolithische Optimierung, was zeigt, dass das Informed Decomposition durchführbar ist. Der Grund, warum es einen Unterschied zwischen beiden Ergebnissen gibt, ist, dass einige Positionen des zulässigen Bereiches wie oben genannt sehr eng sind und der gesamte Designraum noch nicht vollständig erkundet ist. Solche Probleme können die Qualität der Metamodelle beschädigen, sodass durch die Metamodelle die abweichenden Massen erhalten werden.

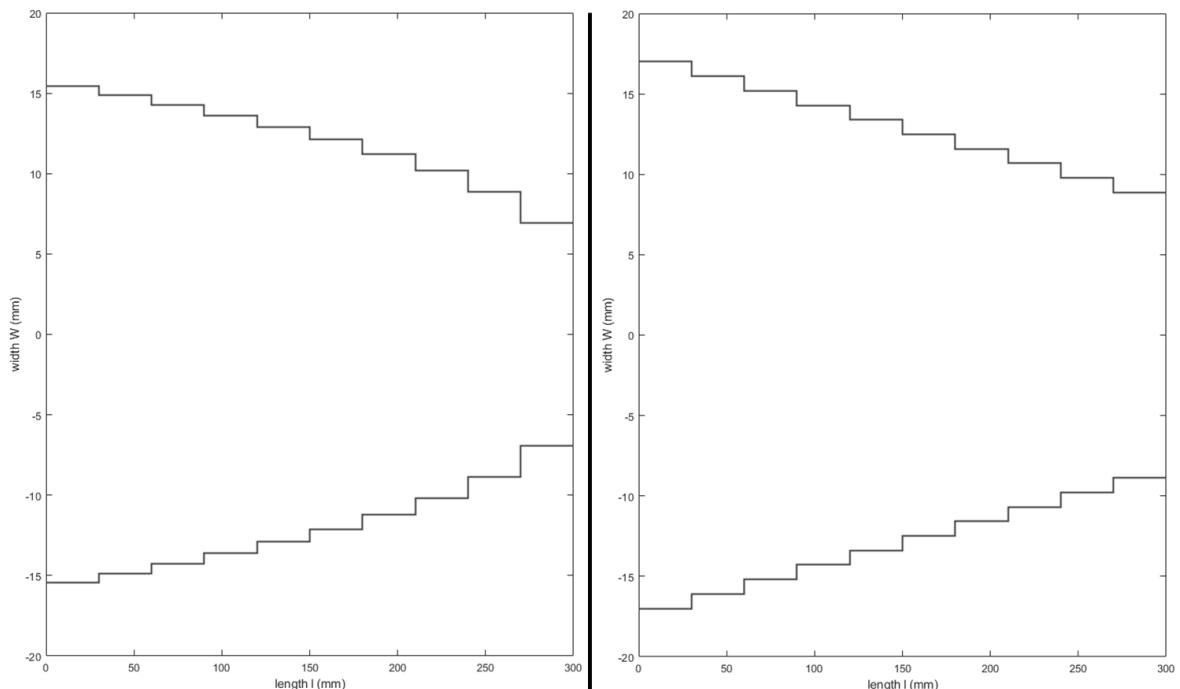
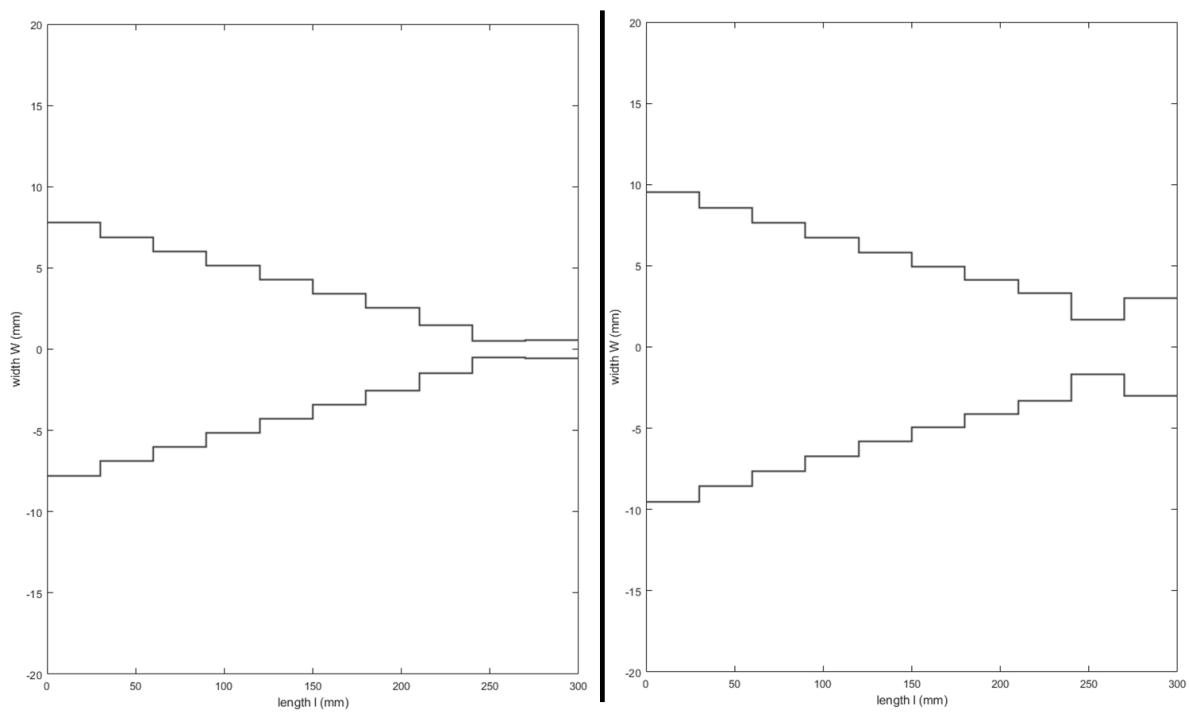


Abb. 4-14: Referenzstruktur von der ersten Komponente nach monolithischer Optimierung (links) sowie nach informed Decomposition (rechts)



*Abb. 4-15: Referenzstruktur von der zweiten Komponente nach monolithischer Optimierung (links) sowie nach informed Decomposition (rechts)*

Die Abbildungen 4-14 und 4-15 zeigen die Referenzstrukturen der zwei Komponenten durch die beiden Optimierungsverfahren. Die Referenzstrukturen können von der endgültigen Ist-Struktur abweichen und ist nur eine Referenz für weiteres Design.

## 5 2D Topologie

Der Prozess der zweidimensionalen topologischen Optimierung ist im Wesentlichen gleich wie das eindimensionale Optimierungsproblem und es sind nur einige Änderungen erforderlich. In diesem Kapitel werden hauptsächlich solche Änderungen vorgestellt und die Ergebnisse der zweidimensionalen Optimierung angegeben.

### 5.1 Problemdefinition

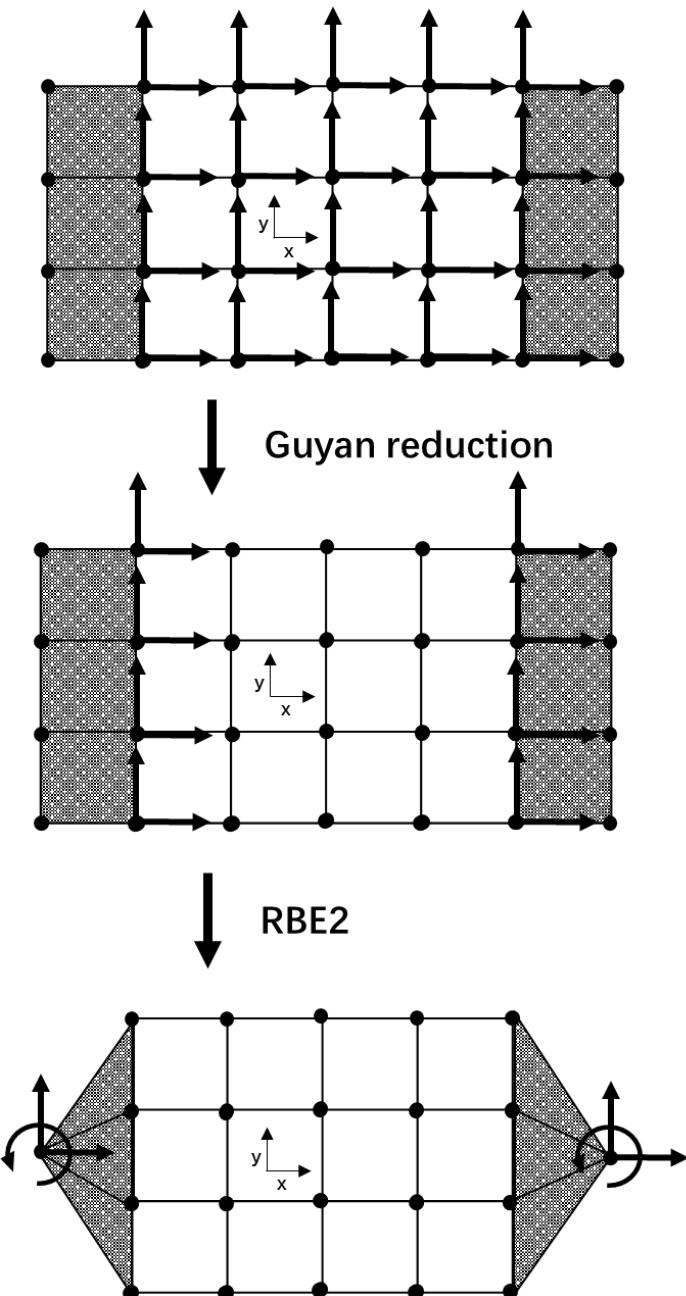


Abb. 5-1: Zu optimierende 2D-Topologie und der Prozess von der Finite-Elemente-Analyse bis zum Erhalten der  $6 \times 6$  Steifigkeitsmatrix

Die im zweidimensionalen Optimierungsproblem verwendete Ausgangsstruktur ist eine einfache Quader-Topologie. Die Länge beträgt 300 mm, die Breite beträgt 100 mm und die Dicke beträgt 1 mm.

Es ist zu beachten, dass die tatsächlich verwendete Länge für die Finite-Elemente-Analyse 0.98 der Gesamtlänge beträgt, um Platz für das Starrkörperteil bei RBE2 zu lassen. Um die einzelnen Finite-Elemente-Elemente quadratisch zu halten, wird auch die verwendete Breite für die Finite-Elemente-Analyse auf 0.98 der Gesamtbreite reduziert.

Bei der zweidimensionalen Finite-Elemente-Analyse wird die Topologie in insgesamt  $30(Länge, n_x) \times 10(Breite, n_y) = 300$  Finite-Elementen unterteilt und die Länge und Breite jedes Elements werden mit  $a$  und  $b$  bezeichnet. Die Designvariable stellt hier die Dichte  $x$  jedes finiten Elements dar. Wenn ein einzelnes finites Element mit vollem Material gefüllt ist, ist die Dichte der Maximalwert eins, nämlich  $x_{max} = 1$ . Im Gegensatz dazu, wenn es ganz leer ist, ist die Dichte der minimale Wert null. An dieser Stelle sei darauf hingewiesen, dass, wie in der Gleichung (2-25) erwähnt wird, das Minimum der Designvariablen für das Strukturoptimierungsproblem einen Wert etwas größer als 0 angenommen werden sollte, daher wird hier angenommen, dass  $x_{min} = 1e - 3$ .

Mit  $x_e$  kann die Steifigkeitsmatrix  $K_e$  von jedem Finite-Element ausgerechnet:

$$K_e = k_e \cdot x_e^p \quad (5 - 1)$$

Jedes finite Element hat 4 Knoten und jeder Knoten hat zwei Freiheitsgrade, d. h. jedes finite Element hat insgesamt 8 Freiheitsgrade, und seine Steifigkeitsmatrix  $k_e$  sollte entsprechend  $8 \times 8$  sein, siehe Anhang 1 für Details.  $x_e$  ist die Dichte von einzelne Finite-Elemente.  $p$  ist ein Straffaktor anhand der Gleichung (2-25) und wird als  $p = 3$  definiert.

Die Steifigkeitsmatrix  $K_e$  kann durch Finite-Elemente-Analyse zu  $K_k$  kombiniert werden.  $K_k$  kann durch Guyan-Reduktion und RBE2 zu einer vereinfachten Steifigkeitsmatrix vereinfacht werden. Da es einen zusätzlichen Schritt RBE2 gibt, ist es notwendig, in der Gleichung (2-12) eine weitere Transformationsmatrix  $T_R$  hinzuzufügen, die in der folgenden Gleichung dargestellt ist (Liu und Quek, 2013, Sek.11.11):

$$K_{RG} = T_R^T \cdot T_G^T \cdot K \cdot T_G \cdot T_R \quad (5 - 2)$$

Weil hierbei die vereinfachte Steifigkeitsmatrix  $K_{RG}$  durch zwei vereinfachte Methode erhalten wird, wird es mit Index  $RG$  markiert, wie in Abbildung 1-2-a) dargestellt ist. Die Struktur, die nach RBE2 erhalten wird, hat zwei unabhängige Knoten, und jeder Knoten hat 3 Freiheitsgrade, daher die vereinfachte Steifigkeitsmatrix die gleiche Größe  $6 \times 6$  wie die im 1D-Strukturoptimierungsproblem besitzt. Dann durch  $\kappa$ -Repräsentation kann der  $\kappa$  - Vektor erhalten werden, der ebenfalls ein Vektor mit vier Elementen ist.

Beim Prozess wie in der Abbildung 3-2-a) wird in diesem Kapitel vor dem Training jedes SVMs 500 neue  $\kappa$ -Vektoren bzw. Samples generiert, d. h. In  $\kappa_1$  sowie  $\kappa_i$  werden je 500 Samples enthalten.

Während des ‚Samples validieren‘ Prozesses muss auch das Optimierungsproblem aufgrund der Änderung von Designvariable neu formuliert werden:

$$\begin{aligned} & \min_x m(x) \\ \text{subject to: } & \frac{|\kappa(x) - \kappa_0|}{\kappa_0} \leq \varepsilon, \quad 0.001 < x_e \leq 1 \end{aligned} \quad (5-3)$$

Die normalisierte Masse  $m(x)$  kann als

$$m(x) = \sum_{e=1}^{n_{ele}} x_e \cdot \frac{100}{n_{ele}} \text{ mit } n_{ele} = n_x \cdot n_y \quad (5-4)$$

formuliert werden.

Die Ableitungen vom wichtigen Parameter für MMA ergibt sich aus der folgenden Formel (O'Connell et al., 1976):

$$\frac{\partial m(x)}{\partial x_e} = \frac{100}{n_{ele}} \quad (5-5)$$

$$\frac{\partial K_{RG}}{\partial x_e} = T_R^T \cdot T_G^T \cdot \frac{\partial K_{Kombiniert}(x)}{\partial x_e} \cdot T_G \cdot T_R \quad (5-6)$$

Es ist zu beachten, dass in 2D- und 3D-Strukturen ein Phänomen, die heißt *Checkerboard Pattern*, auftreten kann. Dieses Phänomen ist ein Vertauschen von den vollen und leeren Finite-Elementen. Die Struktur mit so einem Phänomen kann in der Wirklichkeit nicht realisiert werden, weil die Dichten von zwei benachbarten finiten Elementen nicht kontinuierlich sind. Deshalb eine gewisse Änderung der Ableitung  $\frac{\partial K_k(x)}{\partial x_e}$  erfordert wird (Sigmund, 1994, 1997), um dieses Phänomen zu vermeiden, wie in der folgenden Formel gezeigt:

$$\begin{aligned} \frac{\partial \hat{K}_k(x)}{\partial x_e} &= \frac{1}{x_e \cdot \sum_f^{n_{ele}} \hat{H}_f} \cdot \sum_f^{n_{ele}} \hat{H}_f \cdot x_f \cdot \frac{\partial K_k(x)}{\partial x_e} \\ \hat{H}_f &= r_{min} - \overline{ef}, \quad f \in n_{ele}, \quad \overline{ef} \leq r_{min} \end{aligned} \quad (5-7)$$

Der Abstand zwischen den Mittelpunkten der Elemente  $e$  und  $f$  wird kurz als  $\overline{ef}$  und der Filterradius als  $r_{min}$  angegeben. Die Definition des Faltungsoperators  $\hat{H}_f$  führt zu einem linearen Abfall seines Wertes mit zunehmendem Abstand  $\overline{ef}$  innerhalb des durch  $r_{min}$  definierten Bereichs. Außerhalb dieses Bereichs ist der Faltungsoperator gleich null. In diesem Kapitel wird  $r_{min} = 1.1$  eingestellt.

Alle in diesem Abschnitt aufgeführten wichtigen Parameter sind in der folgenden Tabelle zusammengefasst:

Topologie			Finite Elemente	$\kappa_1$	$\kappa_i$	A	B	$p$	$r_{min}$
Länge	Bereite	Dicke	$n_x = 30$	500	500	5500	6000	3	1.1
300mm	100mm	1mm	$n_y = 10$						

Tabelle 5-1: Wichtige Parameter für die Optimierung von 2D Topologie

## 5.2 Ergebnisse

### 5.2.1 Parameterstudie

Zunächst wie im letzten Kapitel der Einfluss einiges wichtigen Parameters und deren Anpassung zur Optimierung der zweidimensionalen Struktur wird vorgestellt.

#### Epsilon $\varepsilon$

Die Anzahl der Finite Elemente  $n_{ele}$  in zweidimensionaler Struktur ist größer als die in der eindimensionalen Struktur, deshalb ist der zulässige Bereich im Designraum theoretisch „dicker“ als der im eindimensionalen. Um die Rechenzeit zu sparen, kann die Größe von  $\varepsilon$  bei der zweidimensionalen Optimierung sich erhöhen, hierbei wird  $\varepsilon = 1e - 2$  angenommen. Um zu überprüfen, ob eine solche Änderung durchführbar ist, ist eine Validierung erforderlich.

Diese Validierung läuft wie folgende: zunächst wird unter den gleichen äußereren Kraft- und Verschiebungseinschränkung ( $F = 49.03kN, uc = 1mm$  in vertikaler Richtung) wie bei der eindimensionalen Struktur ein optimaler  $\kappa$ -Vektor  $\kappa_{optimal}$  unter Verwendung der monolithischen Optimierung erhalten, wobei  $\kappa_{optimal}$  zulässig sein muss. Als nächstes wird  $\kappa_{optimal}$  in jeder Richtung sich vergrößern oder verkleinern, um einen neuen  $\kappa$ -Vektor  $\kappa_{neu}$  zu erhalten.  $\kappa_{neu}$  werden bei der Verwendung vom gleichen Prozess wie ‚Samples Verifikation‘ verifiziert.  $\varepsilon = 1e - 2$  wird in diesem Prozess eingestellt. Wenn  $\kappa_{neu}$  immer noch zulässig ist, bedeutet dies, dass der zulässige Bereich dick genug ist, um  $\varepsilon = 1e - 2$  zu tolerieren. Der Verhältnisse zwischen  $\kappa_{optimal}$  und  $\kappa_{neu}$  ist wie folgende:

$$\begin{aligned} \kappa_{optimal} &= [\kappa_{optimal,11}, \kappa_{optimal,22}, \kappa_{optimal,33}, \kappa_{optimal,66}]; \\ \kappa_{neu} &= \kappa_{optimal} \text{ mit } \kappa_{neu,i} = g \cdot \kappa_{optimal,i}; \\ g &= \pm 0.05, \pm 0.1 \text{ und } i \in [11, 22, 33, 66] \end{aligned} \quad (5-9)$$

Die verifizierten Ergebnisse sind in der folgenden Tabelle aufgeführt:

$g$	Komponente 1				Komponente 2			
	$\kappa_{neu,11}$	$\kappa_{neu,22}$	$\kappa_{neu,33}$	$\kappa_{neu,66}$	$\kappa_{neu,11}$	$\kappa_{neu,22}$	$\kappa_{neu,33}$	$\kappa_{neu,66}$
+0.1	1	1	-1	-1	1	1	1	1
+0.05	1	1	1	1	1	1	1	1
-0.05	1	1	1	1	1	1	1	1
-0.1	1	-1	1	1	1	1	1	1

Tabelle 5-2: Samples Verifikation für  $\varepsilon$ -Anpassung. „1“ für zulässig und „-1“ für nicht zulässig

Aus der Tabelle ist ersichtlich, dass bei  $g = \pm 0.05$  alle  $\kappa_{neu}$  als „zulässig“ verifiziert werden, was bedeutet, dass  $\varepsilon = 1e - 2$  durchführbar ist. Bei  $g = \pm 0.1$  werden einige  $\kappa_{neu}$  als „unzulässig“ verifiziert, was darauf hinweist, dass ihre entsprechendes  $\kappa_{optimal}$  liegt sehr nahe an der Grenze des zulässigen Bereichs. Die kann aber noch mit  $\varepsilon = 1e - 2$  optimiert werden.

## Ratio

Im vorigen Kapitel wird Ratio schließlich als 1 gewählt. Hier gilt es zu prüfen, ob  $Ratio = 1$  noch für das aktuelle Optimierungsproblem geeignet ist.

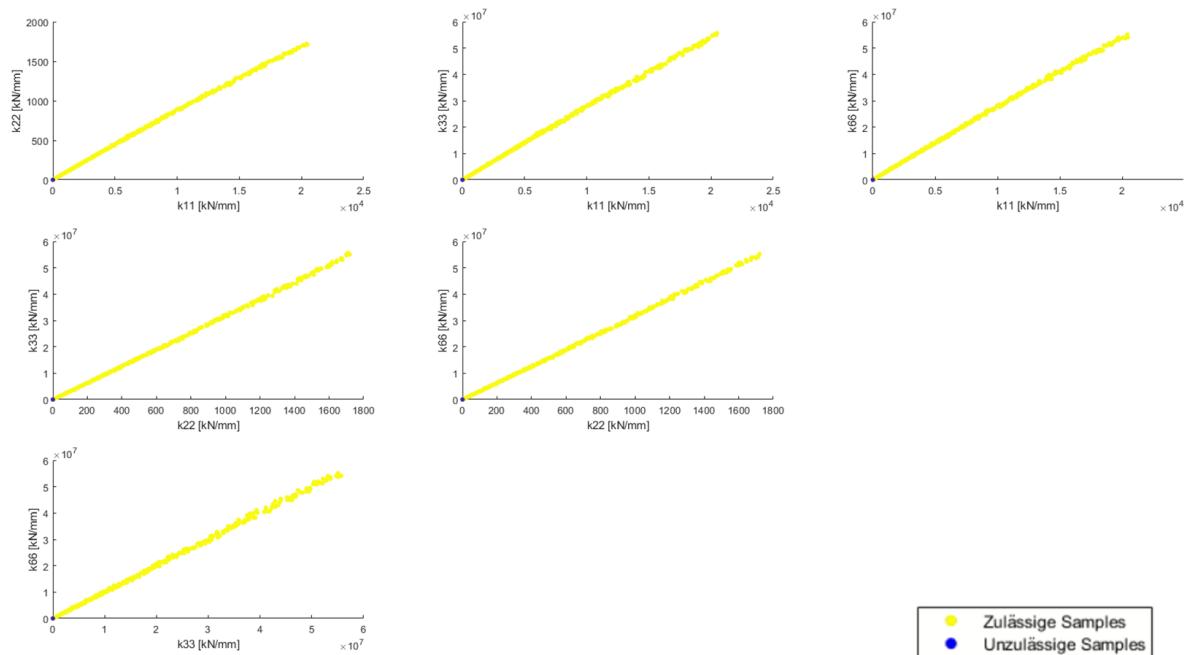


Abb. 5-2: Die Verteilung des ersten 500 Zulässigkeit-Samples mit  $Ratio=0.1$

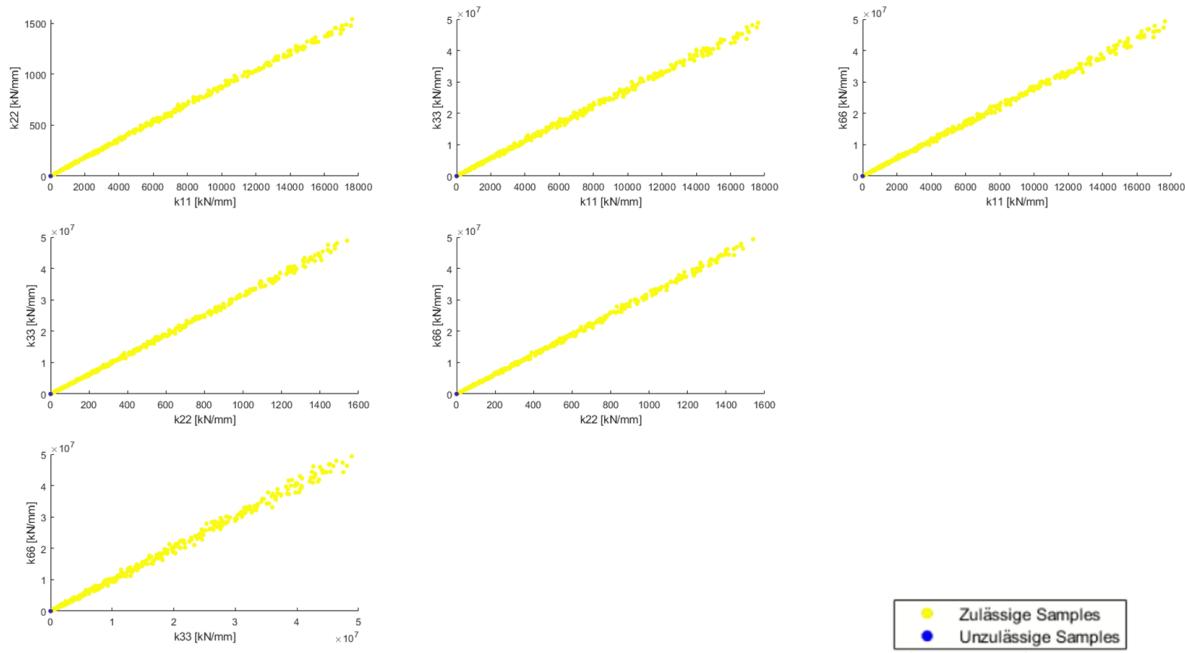


Abb. 5-3: Die Verteilung des ersten 500 Zulässigkeit-Samples mit Ratio=0.2

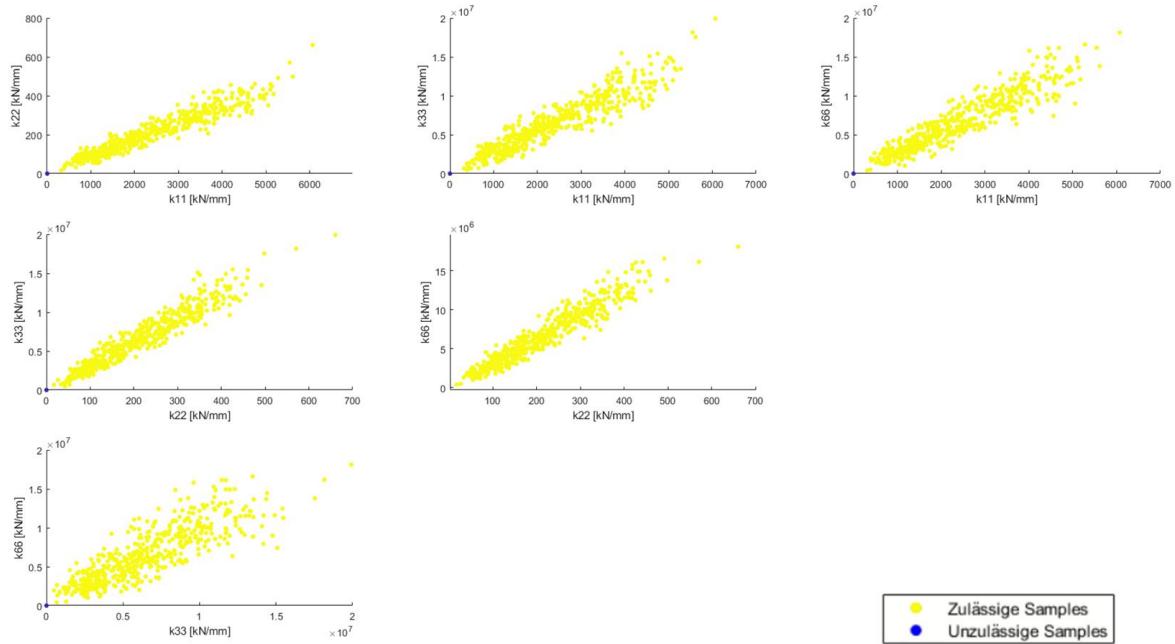


Abb. 5-4: Die Verteilung des ersten 500 Zulässigkeit-Samples mit Ratio=0.75

Anders als im vorherigen Kapitel, wenn die Ratio nahe 1 liegt, wie in Abbildung 5-4 gezeigt, wegen einem viel größeren Designraum ist die Erweiterung des zulässigen Bereichs vom unteren links nach rechts oben im Vergleich zur Abbildung 5-2 und 5-3 offensichtlich nicht

ausreichend, deshalb ist Ratio für das zweidimensionale Strukturoptimierungsproblem nicht mehr 1, sondern unter Berücksichtigung der höhere Explorationsgrade des Designraums wird hier  $Ratio = 0.1$  gewählt.

### $N_{temp}$

Wie im Abschnitt 3.2 erwähnt, wird beim Ausführen von ‚Input Samples generieren‘ eine große Anzahl von Samples zufällig generiert. Die Größe dieser Anzahl in jeder Iteration beeinflusst auch die Endergebnisse. Eine Gleichung wird formuliert, um die Anzahl von zufälligen generiert Samples in jeder Iteration  $N_{temp}$  zu kontrollieren:

$$N_{temp} = \frac{F^i \cdot (j - 10^5)}{(F_{ges} - 1)^i} + 10^5 \quad (5 - 10)$$

$F_{ges}$  ist das Fraction, hierbei wird Fraction gleich wie in letztem Kapitel als 10 definiert.  $F$  ist die Nummer der aktuellen Iteration.  $i$  und  $j$  sind zwei künstliche definierte Konstante. Der Verlauf von  $N_{temp}$  basierend auf  $i$  und  $j$  ist in folgender Abbildung gezeigt:

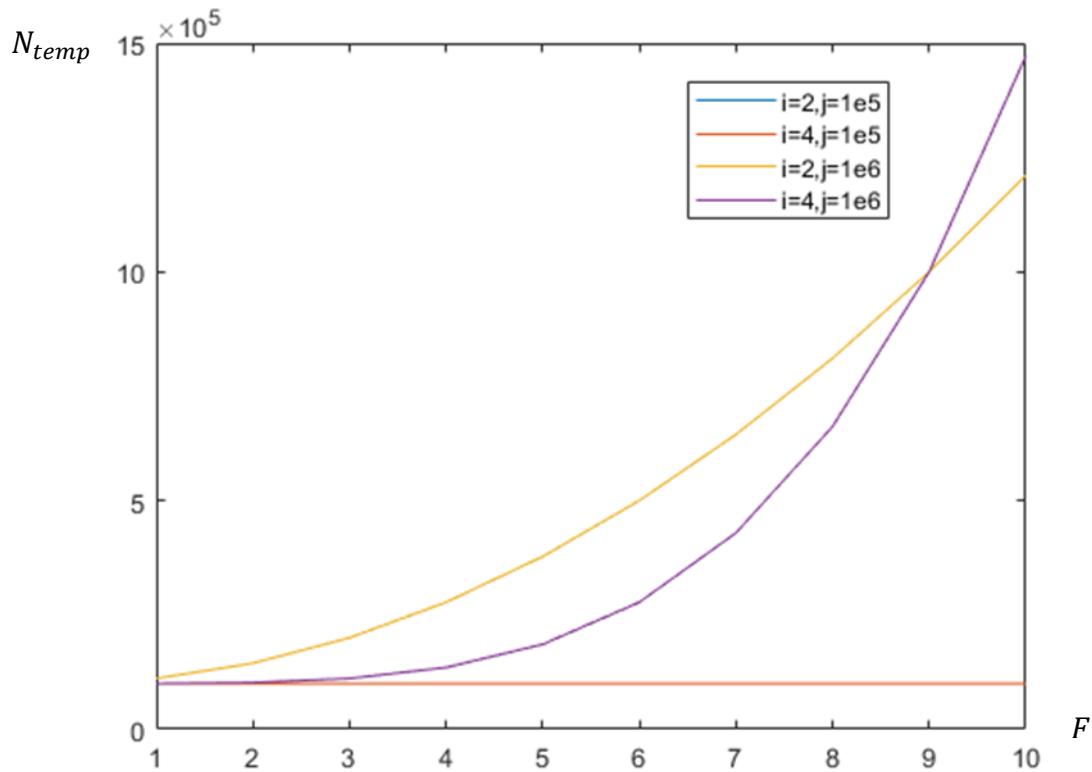


Abb. 5-5: Der Verlauf von  $N_{temp}$  basierend auf  $i$  und  $j$

Je mehr zufällige Samples in einer Iteration generiert werden, desto näher liegen die für ‚Samples validieren‘ ausgewählten Samples an der Hyperebene und desto weniger wird der Designraum in dieser Iteration erkundet. Umgekehrt, je weniger zufällige Samples in einer

Iteration generiert werden, desto größer ist der Erkundungsgrad des Designraums in dieser Iteration. Ein größerer Erkundungsgrad bedeutet auch, dass die ausgewählten Samples in einem größeren Raum verteilt sind und ihre Verteilung enger wird.

Im Experiment ist einer höhere Explorationsgrad bzw. kleiner  $N_{temp}$  in den ersten Fractions erwartet, damit der am Ende erhaltene zulässige Bereich möglichst vollständig sein kann. Die in den letzten Fractions ausgewählten Samples sollen enger verteilt werden, nämlich größer  $N_{temp}$ , sodass der ganze zulässige Bereich eine klare Grenze hat. Deshalb sollen die Konstante  $i$  und  $j$  anhand der Abbildung 5-5 als 4 und  $1e6$  definiert werden, nämlich  $i = 4$  und  $j = 10^6$ .

## Andere Parameter

Im Abschnitt 4.2.1 wird ein Begriff, nämlich negative Massensamples, eingeführt. Dessen Koeffizient wird als 1.2 definiert. In diesem Kapitel wird dieser Koeffizient wegen eines größeren zulässigen Bereichs als 1.1 definiert.

Die obigen Hauptparameter werden in die Tabelle unten zusammengefasst und alle anderen Parameter sind die gleichen wie im vorherigen Kapitel.

Samples Anzahl	$\varepsilon$	Fraction	Negative Massensample	Ratio	$N_{temp}$
5500	1e-2	10	*1.1	0.1	$i = 4$ $j = 10^6$

Tabelle 5-3: Die Werte von den Hauptparametern bei 2D topologischer Optimierung

## 5.2.2 Endergebnisse

Als nächstes werden die Ergebnisse der monolithischen Optimierung und der verteilten Optimierung in der 2D-Strukturoptimierung verglichen.

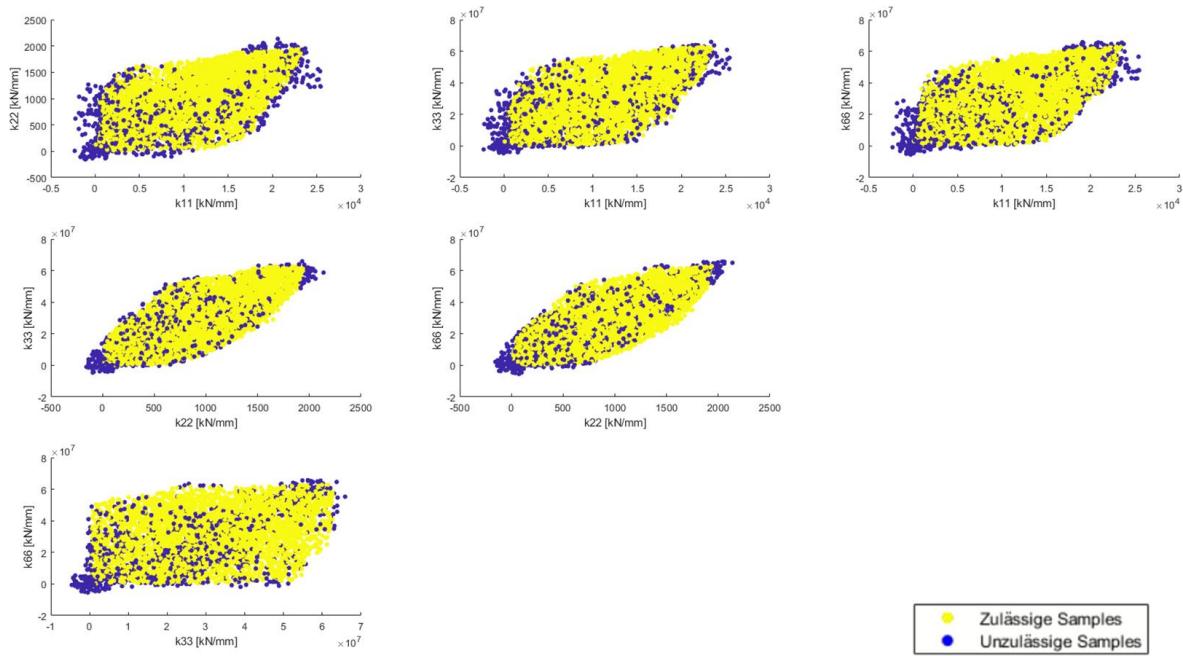


Abb. 5-6: Die Verteilung des gesamten 5500 Zulässigkeit-Samples

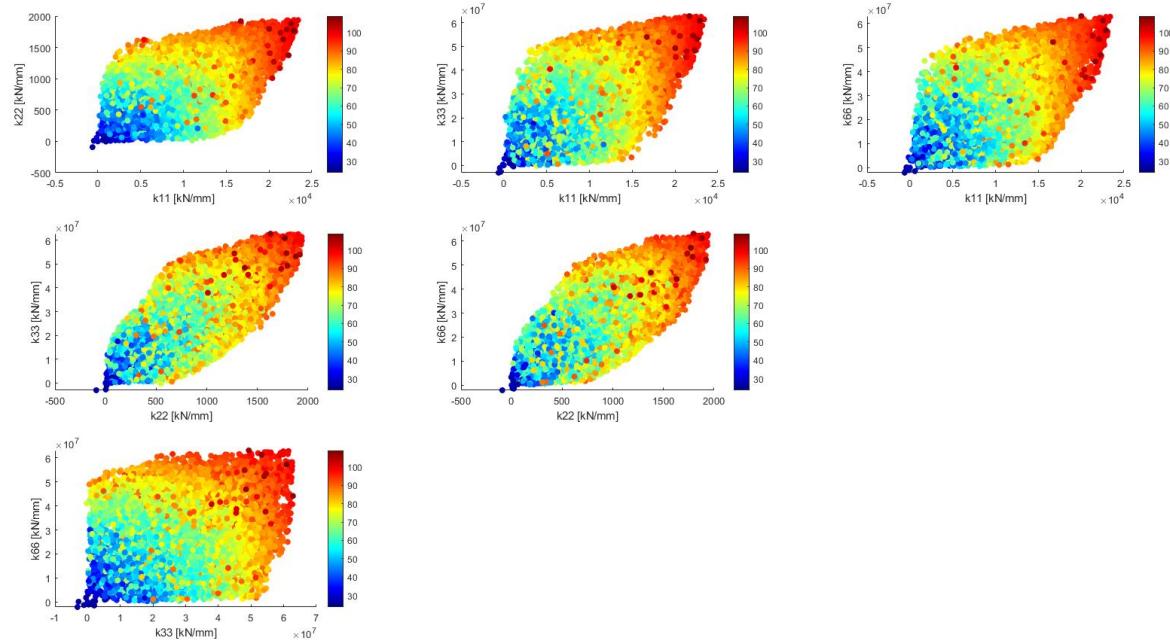


Abb. 5-7: Die Verteilung des gesamten 5500 Masse-Samples (darin keine negativen Massensamples enthält)

Wie aus den Abbildung 5-6 und 5-7 ersichtlich, ist die Verteilung des Samples in zweidimensionalem Strukturoptimierungsproblem im Vergleich zu der in eindimensionalem Strukturoptimierungsproblem gleichmäßiger und die Grenze des zulässigen Bereichs ist ebenfalls deutlicher. Die Verteilung des Samples ist jedoch dünn-besetzter, da die Anzahl des

Samples in Vergleich zum vorherigen Kapitel geringer ist aber der zulässige Bereich größer ist.

Bei zweidimensionalen Optimierungsproblemen ist es notwendig, die Allgemeingültigkeit des Verfahrens Informed Decomposition für unterschiedliche Belastungsbedingungen zu testen. Diese Arbeit bietet 5 verschiedene Belastungsbedingungen, siehe Tabelle 5-4, wobei „ $y_l$ “ sich auf die Belastungsbedingung bezieht, bei der das linke Ende der linken Komponente das freie Ende und das rechte Ende der rechten Komponente mit der Wand verbunden ist.

Lastfall	1	2	3	4	5
Kraft [kN]	49.03	49.03*5	49.03*400	49.03*100000	49.03
Krafrichtung (x/y/r/ $y_l$ )	y	x	x	r	$y_l$

Tabelle 5-4: 5 Lastfallen mit der Größe der Kraft und der Krafrichtung. „x“ für normale Kraft; „y“ für vertikale Kraft; „r“ für Rotation und „ $y_l$ “ für vertikale Kraft auf der anderen Seite der Komponente

		Erste Komponente		Zweite Komponente	
		$\kappa_{1,optimal}$	$m_{1,optimal}$	$\kappa_{2,optimal}$	$m_{2,optimal}$
Lastfall 1	Monolithische Optimierung	$\begin{bmatrix} 9.9297e + 03 \\ 422.6876 \\ 2.6010e + 07 \\ 2.2324e + 07 \end{bmatrix}$	57.7815	$\begin{bmatrix} 1.8796e + 03 \\ 199.0337 \\ 1.6838e + 07 \\ 4.9175e + 05 \end{bmatrix}$	39.1609
	Informed Decomposition	$\begin{bmatrix} 8.7502e + 03 \\ 404.0366 \\ 2.6274e + 07 \\ 1.8212e + 07 \end{bmatrix}$	55.2039	$\begin{bmatrix} 3.3199e + 03 \\ 246.0695 \\ 1.9482e + 07 \\ 1.3590e + 06 \end{bmatrix}$	41.8000
Lastfall 2	Monolithische Optimierung	$\begin{bmatrix} 485.5156 \\ 1.5957 \\ 6.4270e + 04 \\ 6.4273e + 04 \end{bmatrix}$	20.2275	$\begin{bmatrix} 485.5156 \\ 1.5957 \\ 6.4270e + 04 \\ 6.4273e + 04 \end{bmatrix}$	20.2275
	Informed Decomposition	$\begin{bmatrix} 2.1331e + 03 \\ 6.7222 \\ 1.3199e + 06 \\ 1.7573e + 06 \end{bmatrix}$	25.3621	$\begin{bmatrix} 2.1954e + 03 \\ 6.3419 \\ 1.4197e + 06 \\ 1.7610e + 06 \end{bmatrix}$	26.5483
Lastfall 3	Monolithische Optimierung	$\begin{bmatrix} 2.3347e + 04 \\ 1.9452e + 03 \\ 6.2444e + 07 \\ 6.2444e + 07 \end{bmatrix}$	99.6510	$\begin{bmatrix} 2.3347e + 04 \\ 1.9452e + 03 \\ 6.2444e + 07 \\ 6.2444e + 07 \end{bmatrix}$	99.6510
	Informed Decomposition	$\begin{bmatrix} 2.3424e + 04 \\ 1.9290e + 03 \\ 6.2187e + 07 \\ 6.1780e + 07 \end{bmatrix}$	99.7579	$\begin{bmatrix} 2.3424e + 04 \\ 1.9291e + 03 \\ 6.2187e + 07 \\ 6.1780e + 07 \end{bmatrix}$	99.7579

Lastfall 4	Monolithische Optimierung	$\begin{bmatrix} 9.7261e + 03 \\ 264.8996 \\ 7.4350e + 06 \\ 7.4350e + 06 \end{bmatrix}$	49.2105	$\begin{bmatrix} 9.7261e + 03 \\ 264.8996 \\ 7.4350e + 06 \\ 7.4350e + 06 \end{bmatrix}$	49.2105
	Informed Decomposition	$\begin{bmatrix} 9.7309e + 03 \\ 291.5527 \\ 1.6431e + 07 \\ 3.8271e + 06 \end{bmatrix}$	53.2800	$\begin{bmatrix} 9.6658e + 03 \\ 284.3684 \\ 1.6654e + 07 \\ 3.7821e + 06 \end{bmatrix}$	52.5533
Lastfall 5	Monolithische Optimierung	$\begin{bmatrix} 1.8794e + 03 \\ 199.0308 \\ 4.9170e + 05 \\ 1.6838e + 07 \end{bmatrix}$	39.1615	$\begin{bmatrix} 9.9307e + 03 \\ 422.6736 \\ 2.2325e + 07 \\ 2.6011e + 07 \end{bmatrix}$	57.7861
	Informed Decomposition	$\begin{bmatrix} 2.4120e + 03 \\ 208.0058 \\ 1.0856e + 06 \\ 1.6721e + 07 \end{bmatrix}$	39.5148	$\begin{bmatrix} 9.3467e + 03 \\ 550.9149 \\ 2.3226e + 07 \\ 2.9542e + 07 \end{bmatrix}$	58.8709

Tabelle 5-5: Die optimalen Parameter von den zwei Komponenten nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfallen

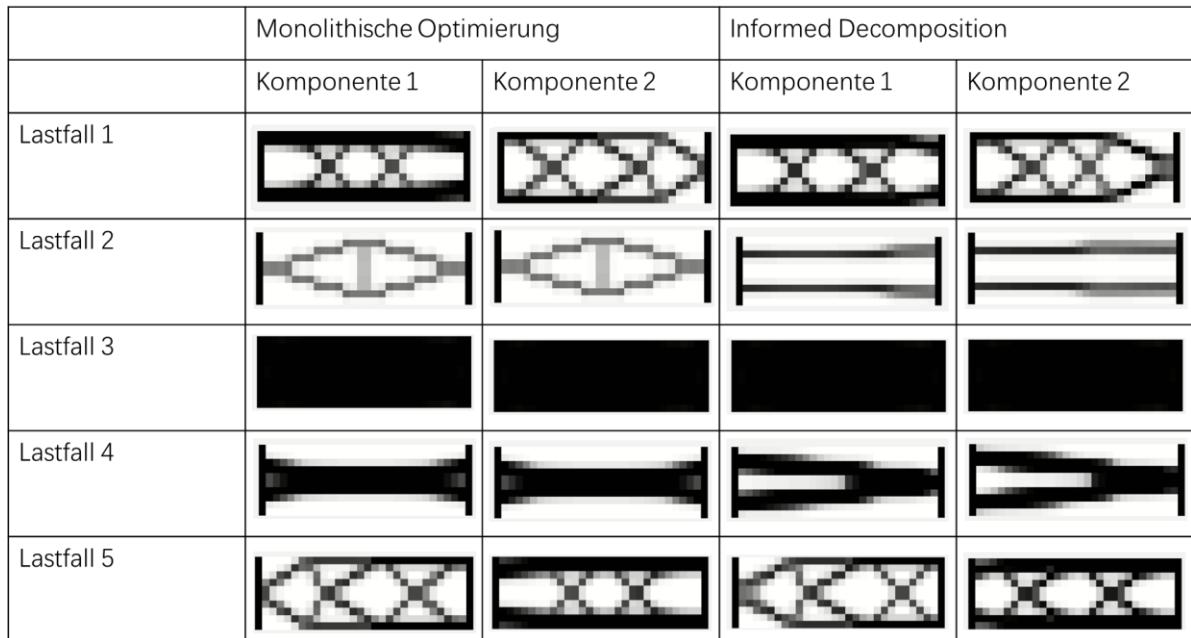


Abb. 5-8: Referenzstruktur von der ersten Komponente nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfallen

Tabelle 5-5 vergleichen die Ergebnisse der beiden Optimierungsverfahren unter fünf Belastungsbedingungen. Die optimale Gesamtmasse ( $m_1 + m_2$ ) durch das Informed Decomposition ist auch etwas größer als die durch das monolithische Optimierungsverfahren, was zeigt, dass das Informed Decomposition in zwei Dimension auch durchführbar ist. Es

gibt Unterschiede in den Ergebnissen der beiden Verfahren, die durch verschiedene Toleranzen, der Parameterauswahl, der Genauigkeit des Metamodells etc. beim Informed Decomposition bedingt sein können. Die Genauigkeit der vorliegenden Ergebnisse ist aber schon akzeptabel.

Abbildung 5-8 stellen die Referenzstrukturen der beiden Optimierungsverfahren durch Komponentenoptimierung dar. Wie am Ende des vorherigen Kapitels erwähnt, dienen diese Strukturen nur als Referenz und können von der tatsächlichen Struktur abweichen.

## 6 3D Topologie

In diesem Kapitel wird dreidimensionale Topologie optimiert, wobei die meisten Methoden und Parameter auf den Erfahrungen mit zweidimensionalen Strukturen basieren. Es gibt darin nur wenige Änderungen.

### 6.1 Problemdefinition

Die für die 3D-Optimierung verwendete Topologie ist dieselbe wie bei 2D, aber bei der 3D Finite-Elemente-Analyse sollte es in der Dickenrichtung auch in einigen Finite-Elementen unterteilt werden, wie in der folgenden Figure gezeigt:

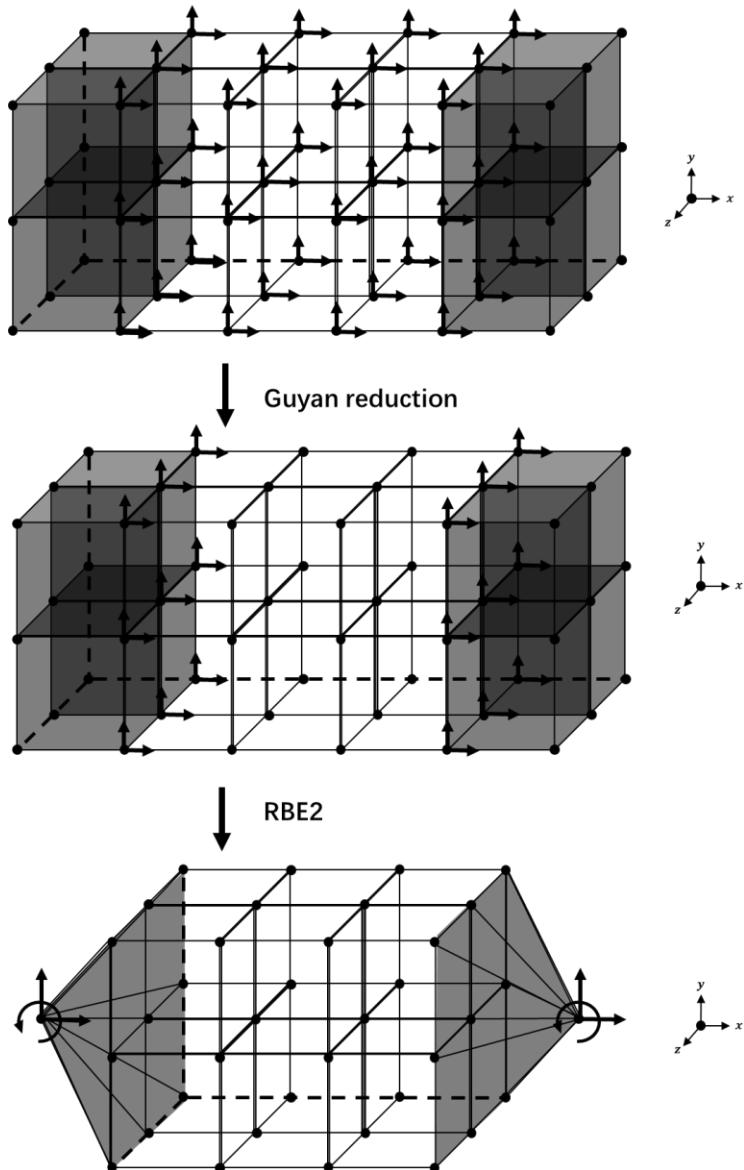


Abb. 6-1: Zu optimierende 3D-Topologie und der Prozess von der Finite-Elemente-Analyse bis zum Erhalten der  $6 \times 6$  Steifigkeitsmatrix

Die Länge der Komponente beträgt 300 mm, die Breite ist 75 mm und die Höhe ist 25 mm. Die Anzahl der finiten Elemente, die in drei Richtungen aufgeteilt werden, beträgt  $n_x = 30$ ,  $n_y = 10$ ,  $n_z = 4$ . Die Designvariable  $x$  ist gleich wie im fünften Kapitel die Dichte des einzelnen Finite-Elements und ihr Wertebereich entspricht dem von 2D, nämlich  $x_{max} = 1$  und  $x_{min} = 0.001$ . Straffaktor  $p$  in der Gleichung (5-1) wird hier als 4 definiert, nämlich  $p = 4$ . Außerdem hat das dreidimensionale Finite Element 8 Knoten und 16 Freiheitsgrade, sodass die Steifigkeitsmatrix für dreidimensionale Steifigkeitsmatrix erneut ausgerechnet werden soll. Dazu wird der von Petrarola (2021) bereitgestellte Code verwendet.

$r_{min}$  in der Gleichung (5-7) wird hierbei als  $\sqrt{2}$  definiert, um die Struktur der Finite-Elemente anzupassen.

Alle in diesem Abschnitt aufgeführten wichtigen Parameter sind in der folgenden Tabelle zusammengefasst:

Topologie			Finite Elemente	$\kappa_1$	$\kappa_i$	A	B	$p$	$r_{min}$
Länge	Breite	Höhe	$n_x = 30$ $n_y = 10$ $n_z = 10$	500	500	5500/ 7500	6000/ 8000	4	$\sqrt{2}$
300mm	75mm	25mm							

Tabelle 6-1: Wichtige Parameter für die Optimierung von 3D Topologie

## 6.2 Ergebnisse

### 6.2.1 Parameterstudie

Die meisten im 2D-Optimierungsproblem gesetzten Variablen bleiben im 3D-Optimierungsproblem unverändert. Nur ein Parameter in der Gleichung (5-3) soll hierbei verändert werden.

$$(\kappa_{max} - \kappa_{min})$$

Die Beschränkung auf den  $\kappa$ -Vektor in der Gleichung (5-3), d. h. die erste Beschränkung, dient hauptsächlich dazu, einen  $\kappa$ -Vektor  $\kappa(x)$  zu finden, der dem Referenz- $\kappa$ -Vektor  $\kappa_0$  so nahe wie möglich kommt. Der  $\kappa$ -Vektor  $\kappa_0$  auf dem Nenner der linken Seite der Gleichung ist eine Konstante und dient dazu, die vier Elementen im Fehler der beiden  $\kappa$ -Vektoren auf dem Zähler, nämlich  $|\kappa(x) - \kappa_0|$ , innerhalb einer gleichen Größenordnung zu halten. Der Vorteil dieses Entwurfs besteht darin, dass der Fehler jedes Elements zwischen beiden  $\kappa$ -Vektoren auf gleicher Stufe verglichen werden kann.

Allerdings ist für manche Konstruktionen, das heißt manche Belastungsbedingung, weil der

Wert von einem Element in  $\kappa_0$  zu klein und der Quotient des Fehlers zwischen beiden  $\kappa$ -Vektoren um dieses Element zu groß sein könnte, sodass der gesamte Optimierungsprozess in lokale Konvergenz fällt und kann nicht die richtigen Ergebnisse erhalten.

Um die Vorteile dieses Entwurfs beizubehalten und gleichzeitig die oben genannten Probleme zu vermeiden, wird das  $\kappa_0$  auf dem Nenner durch einen anderen Konstant ( $\kappa_{max} - \kappa_{min}$ ) ersetzt. Das neue Optimierungsproblem ist wie unten gezeigt:

$$\begin{aligned} & \min_x m(x) \\ & \text{subject to: } \frac{|\kappa(x) - \kappa_0|}{(\kappa_{max} - \kappa_{min})} \leq \varepsilon; \quad 0.001 < x_e \leq 1 \end{aligned} \quad (6-1)$$

In  $(\kappa_{max} - \kappa_{min})$  ist der Wert von jedem Element groß genug, und es gibt einen Unterschied der Größenordnung zwischen den vier Elementen. Durch diese Änderung kann ein besseres Endergebnis erzielt werden.

### 6.2.2 Endergebnisse

Wie aus den Abbildungen 6-2 und 6-3 ersichtlich ist, unterscheidet sich die Verteilung von Samples in der dreidimensionalen Struktur nicht wesentlich von der in der zweidimensionalen Struktur. Weil der zulässige Bereich der dreidimensionalen Struktur größer als der der zweidimensionalen Struktur ist, ist die Verteilung der Samples ein bisschen dünn-besetzter, d. h. mehr Hohlräume im zulässigen Bereich.

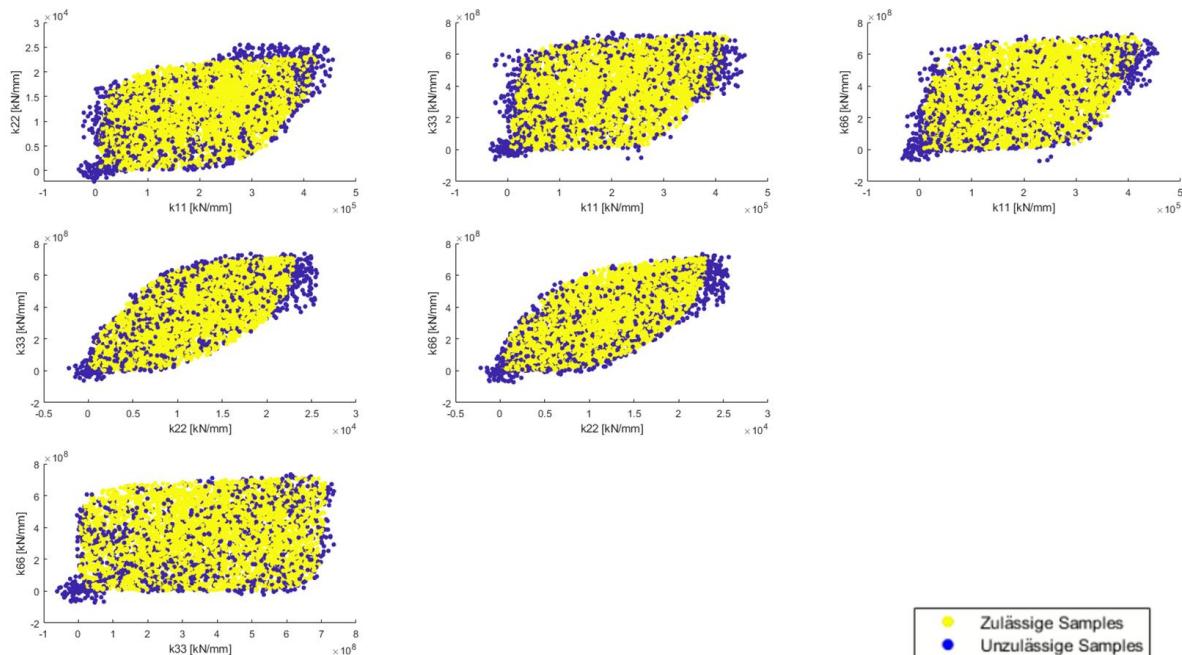


Abb. 6-2: Die Verteilung des gesamten 5500 Zulässigkeit-Samples

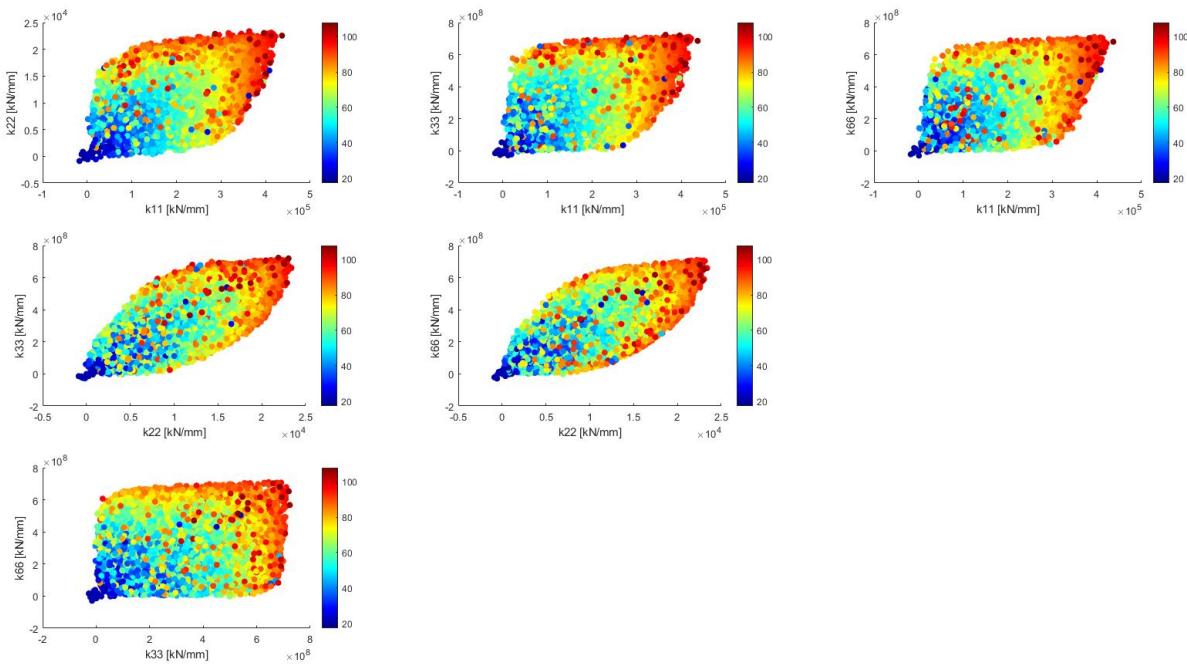


Abb. 6-3: Die Verteilung des gesamten 5500 Masse-Samples (darin keine negativen Massensamples enthält)

Tabelle 6-2 stellt die fünf in diesem Kapitel verwendeten Belastungsbedingungen dar. Da die Dicke der beim 3D-Strukturoptimierungsproblem verwendeten Struktur größer ist, ist es notwendig, eine größere externe Kraft bereitzustellen.

Lastfall	1	2	3	4	5
Kraft [kN]	588.36	588.36*100	588.36*400	588.36*100000	588.36
Kraftrichtung (x/y/r/y <sub>l</sub> )	y	x	x	r	y <sub>l</sub>

Tabelle 6-2: 5 Lastfallen mit der Größe von der Kraft und der Kraftrichtung. „x“ für normale Kraft; „y“ für vertikale Kraft; „r“ für Moment und „y<sub>l</sub>“ für vertikale Kraft auf der anderen Seite der Komponente

		Erste Komponente		Zweite Komponente		
		$\kappa_{1,optimal}$	$m_{1,optimal}$	$\kappa_{2,optimal}$	$m_{2,optimal}$	
Lastfall	1	Monolithische Optimierung	$\begin{bmatrix} 2.1111e + 05 \\ 7.6932e + 03 \\ 3.5002e + 08 \\ 3.2170e + 08 \end{bmatrix}$	55.1398	$\begin{bmatrix} 4.3942e + 04 \\ 2.6743e + 03 \\ 2.1296e + 08 \\ 1.7454e + 07 \end{bmatrix}$	30.6296

	Informed Decomposition	$\begin{bmatrix} 1.9703e + 05 \\ 8.6305e + 03 \\ 3.7314e + 08 \\ 3.1770e + 08 \end{bmatrix}$	54.0545	$\begin{bmatrix} 8.0308e + 04 \\ 5.4333e + 03 \\ 2.5931e + 08 \\ 1.1505e + 07 \end{bmatrix}$	33.1405
Lastfall 2	Monolithische Optimierung	$\begin{bmatrix} 1.1234e + 05 \\ 4.3203e + 03 \\ 1.1461e + 08 \\ 1.1461e + 08 \end{bmatrix}$	34.5853	$\begin{bmatrix} 1.1432e + 05 \\ 4.5272e + 03 \\ 1.1986e + 08 \\ 1.1986e + 08 \end{bmatrix}$	35.1310
	Informed Decomposition	$\begin{bmatrix} 1.1326e + 05 \\ 4.2102e + 03 \\ 1.1579e + 08 \\ 1.5569e + 08 \end{bmatrix}$	35.2636	$\begin{bmatrix} 1.1314e + 05 \\ 4.2190e + 03 \\ 1.1600e + 08 \\ 1.5578e + 08 \end{bmatrix}$	35.1910
Lastfall 3	Monolithische Optimierung	$\begin{bmatrix} 4.3098e + 05 \\ 2.3229e + 04 \\ 7.1943e + 08 \\ 7.1943e + 08 \end{bmatrix}$	99.6349	$\begin{bmatrix} 4.3098e + 05 \\ 2.3229e + 04 \\ 7.1943e + 08 \\ 7.1943e + 08 \end{bmatrix}$	99.6349
	Informed Decomposition	$\begin{bmatrix} 4.3063e + 05 \\ 2.1931e + 04 \\ 6.7427e + 08 \\ 6.9546e + 08 \end{bmatrix}$	99.7364	$\begin{bmatrix} 4.3063e + 05 \\ 2.1930e + 04 \\ 6.7426e + 08 \\ 6.9542e + 08 \end{bmatrix}$	99.7362
Lastfall 4	Monolithische Optimierung	$\begin{bmatrix} 1.1539e + 05 \\ 353.6109 \\ 1.1891e + 08 \\ 1.1891e + 08 \end{bmatrix}$	33.8988	$\begin{bmatrix} 1.1539e + 05 \\ 353.6109 \\ 1.1891e + 08 \\ 1.1891e + 08 \end{bmatrix}$	33.8988
	Informed Decomposition	$\begin{bmatrix} 1.2190e + 05 \\ 4.4279e + 03 \\ 2.1311e + 08 \\ 2.2176e + 08 \end{bmatrix}$	38.3780	$\begin{bmatrix} 1.1714e + 05 \\ 4.0431e + 03 \\ 2.0251e + 08 \\ 2.0932e + 08 \end{bmatrix}$	37.5837
Lastfall 5	Monolithische Optimierung	$\begin{bmatrix} 4.3942e + 04 \\ 2.6743e + 03 \\ 1.7454e + 07 \\ 2.1296e + 08 \end{bmatrix}$	30.6296	$\begin{bmatrix} 2.1111e + 05 \\ 7.6932e + 03 \\ 3.2170e + 08 \\ 3.5002e + 08 \end{bmatrix}$	55.1398
	Informed Decomposition	$\begin{bmatrix} 8.9047e + 04 \\ 4.9277e + 03 \\ 1.1092e + 08 \\ 2.5610e + 08 \end{bmatrix}$	34.3266	$\begin{bmatrix} 1.9238e + 05 \\ 8.1107e + 03 \\ 3.1310e + 08 \\ 3.5684e + 08 \end{bmatrix}$	53.2557

Tabelle 6-3: Die optimalen Parameter von den zwei Komponenten nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfallen

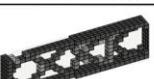
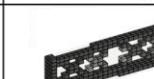
	Monolithische Optimierung		Informed Decomposition	
	Komponente 1	Komponente 2	Komponente 1	Komponente 2
Lastfall 1				
Lastfall 2				
Lastfall 3				
Lastfall 4				
Lastfall 5				

Abb. 6-4: Referenzstruktur von der ersten Komponente nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfällen

Tabelle 6-3 vergleichen die Ergebnisse der beiden Optimierungsverfahren unter fünf Belastungsbedingungen. Die optimale Gesamtmasse ( $m_1 + m_2$ ) durch das Informed Decomposition ist auch wie vorher etwas größer als die durch das monolithische Optimierungsverfahren, was zeigt, dass das Informed Decomposition in drei Dimension auch durchführbar ist. Es gibt noch Unterschiede in den Ergebnissen der beiden Verfahren, aber im Vergleich zum vorherigen Kapitel ist der Unterschied hier geringer, was darauf hindeutet, dass die Methode Informed Decomposition besser auf 3D-Strukturoptimierungsprobleme anwendbar ist.

Abbildung 6-4 stellen die Referenzstrukturen der beiden Optimierungsverfahren durch Komponentenoptimierung dar. Wie am Ende des vorherigen Kapitels erwähnt, dienen diese Strukturen nur als Referenz und können von der tatsächlichen Struktur abweichen.

Bei der Behandlung von dreidimensionalen Strukturoptimierungsproblemen wird in dieser Arbeit noch erwogen, die Anzahl des Samples zu erhöhen, um bessere Ergebnisse zu erzielen. In dem neuen Experiment wird die Gesamtzahl der Samples auf 7500 erhöht. Die Ergebnisse sind wie folgt:

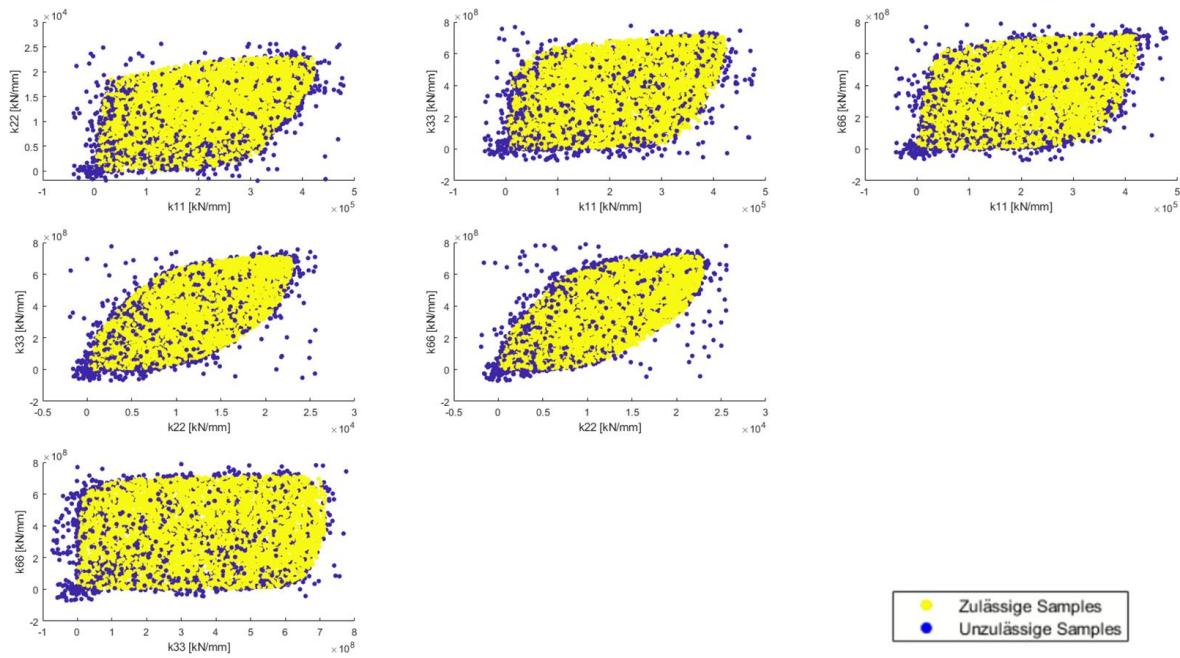


Abb. 6-5: Die Verteilung des gesamten 7500 Zulässigkeit-Samples

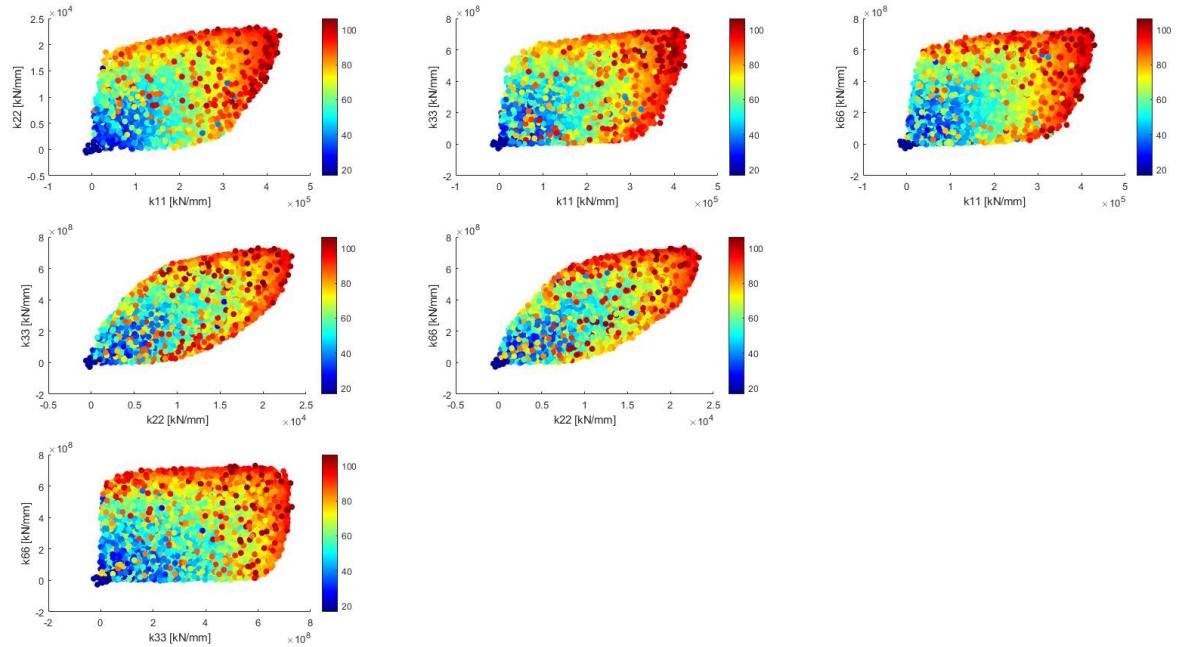


Abb. 6-6: Die Verteilung des gesamten 5500 Masse-Samples (darin keine negativen Massensamples enthält)

		Erste Komponente		Zweite Komponente	
		$\kappa_{1,optimal}$	$m_{1,optimal}$	$\kappa_{2,optimal}$	$m_{2,optimal}$
Lastfall 1	Monolithische Optimierung	$\begin{bmatrix} 2.1111e + 05 \\ 7.6932e + 03 \\ 3.5002e + 08 \\ 3.2170e + 08 \end{bmatrix}$	55.1398	$\begin{bmatrix} 4.3942e + 04 \\ 2.6743e + 03 \\ 2.1296e + 08 \\ 1.7454e + 07 \end{bmatrix}$	30.6296
	Informed Decomposition	$\begin{bmatrix} 1.9755e + 05 \\ 9.0228e + 03 \\ 3.8389e + 08 \\ 3.2853e + 08 \end{bmatrix}$	53.8905	$\begin{bmatrix} 5.6598e + 04 \\ 3.5632e + 03 \\ 2.3753e + 08 \\ 2.8195e + 07 \end{bmatrix}$	32.4098
Lastfall 2	Monolithische Optimierung	$\begin{bmatrix} 1.1234e + 05 \\ 4.3203e + 03 \\ 1.1461e + 08 \\ 1.1461e + 08 \end{bmatrix}$	34.5853	$\begin{bmatrix} 1.1432e + 05 \\ 4.5272e + 03 \\ 1.1986e + 08 \\ 1.1986e + 08 \end{bmatrix}$	35.1310
	Informed Decomposition	$\begin{bmatrix} 1.2222e + 05 \\ 4.8078e + 03 \\ 2.4541e + 08 \\ 8.0422e + 07 \end{bmatrix}$	37.1433	$\begin{bmatrix} 1.0538e + 05 \\ 2.2322e + 03 \\ 3.0983e + 07 \\ 1.4769e + 08 \end{bmatrix}$	33.0470
Lastfall 3	Monolithische Optimierung	$\begin{bmatrix} 4.3098e + 05 \\ 2.3229e + 04 \\ 7.1943e + 08 \\ 7.1943e + 08 \end{bmatrix}$	99.6349	$\begin{bmatrix} 4.3098e + 05 \\ 2.3229e + 04 \\ 7.1943e + 08 \\ 7.1943e + 08 \end{bmatrix}$	99.6349
	Informed Decomposition	$\begin{bmatrix} 4.3036e + 05 \\ 2.1854e + 04 \\ 6.8099e + 08 \\ 6.8461e + 08 \end{bmatrix}$	99.7095	$\begin{bmatrix} 4.3038e + 05 \\ 2.1860e + 04 \\ 6.8158e + 08 \\ 6.8435e + 08 \end{bmatrix}$	99.7104
Lastfall 4	Monolithische Optimierung	$\begin{bmatrix} 1.1539e + 05 \\ 353.6109 \\ 1.1891e + 08 \\ 1.1891e + 08 \end{bmatrix}$	33.8988	$\begin{bmatrix} 1.1539e + 05 \\ 353.6109 \\ 1.1891e + 08 \\ 1.1891e + 08 \end{bmatrix}$	33.8988
	Informed Decomposition	$\begin{bmatrix} 1.2054e + 05 \\ 2.6673e + 03 \\ 1.7705e + 08 \\ 1.7970e + 08 \end{bmatrix}$	36.8873	$\begin{bmatrix} 1.2236e + 05 \\ 2.7508e + 03 \\ 1.7504e + 08 \\ 1.8149e + 08 \end{bmatrix}$	36.7360
Lastfall 5	Monolithische Optimierung	$\begin{bmatrix} 4.3942e + 04 \\ 2.6743e + 03 \\ 1.7454e + 07 \\ 2.1296e + 08 \end{bmatrix}$	30.6296	$\begin{bmatrix} 2.1111e + 05 \\ 7.6932e + 03 \\ 3.2170e + 08 \\ 3.5002e + 08 \end{bmatrix}$	55.1398
	Informed Decomposition	$\begin{bmatrix} 6.2094e + 04 \\ 3.1319e + 03 \\ 2.8376e + 07 \\ 2.3463e + 08 \end{bmatrix}$	34.3266	$\begin{bmatrix} 1.9578e + 05 \\ 8.7406e + 03 \\ 3.2487e + 08 \\ 3.7467e + 08 \end{bmatrix}$	53.5277

Tabelle 6-4: Die optimalen Parameter von den zwei Komponenten nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfällen

	Monolithische Optimierung		Informed Decomposition	
	Komponente 1	Komponente 2	Komponente 1	Komponente 2
Lastfall 1				
Lastfall 2				
Lastfall 3				
Lastfall 4				
Lastfall 5				

Abb. 6-7: Referenzstruktur von der ersten Komponente nach monolithischer Optimierung sowie nach Informed Decomposition für alle 5 Lastfallen

Wie in den Abbildungen 6-5 und 6-6 zu sehen ist, mit den zusätzlichen 2000 Samples gibt es in dem zulässigen Bereich weniger Lücke. Einige unzulässige Samples sind weit vom zulässigen Bereich entfernt, was durch unzureichende zulässige Samples, die in einer bestimmten Iteration durch den Prozess ‚Input Samples generieren‘ generiert werden, verursacht werden kann, dies hat jedoch keinen Einfluss auf das Training des Metamodells.

Aus Tabelle 6-3 ist ersichtlich, dass die Ergebnisse vom Versuch mit 7500 Samples für alle Belastungsbedingungen besser als die vom Versuch mit 5500 Samples sind. Der  $\kappa$ -Vektor ist näher am optimierten  $\kappa$ -Vektor, der durch monolithische Optimierung erhalten wird, und die Gesamtmasse ist auch näher an der durch monolithische Optimierung erhalten Masse. Mit so einer Tendenz kann vorhergesagt werden, dass die Methode Informed Decomposition eine bessere Leistung erbringt, wenn die Gesamtzahl der Samples weiter ansteigt. Durch die Figure 6-7 ist es intuitiver sein, dass die Referenzstruktur unter 7500 Samples auch ähnlicher an der durch monolithisch optimierten Referenzstruktur ist.

## 7 Diskussion

Aus den Endergebnissen der Kapitel 4, 5 und 6 geht hervor, dass die 1D-, 2D- und 3D-topologische Optimierung mittels Informed Decomposition stets konvergente Ergebnisse bzw.  $\kappa_{optimal}$  und  $m_{optimal}$  liefern kann, die den Systemanforderungen genügen. Das bedeutet, dass dieser von Krischer (2021) entwickelte Ansatz für die strukturierte Optimierung einer Topologie mit drei Freiheitsgraden und zwei Interfaces zulässig ist. Diese Schlussfolgerung wird auch durch den Vergleich der via Informed Decomposition erzielten Optimierungsergebnisse mit denen der monolithischen Optimierung bewiesen: Die mittels Ersterem erzielte Optimierungsmasse fällt stets größer als die andere aus, was bedeutet, dass die Optimierungsergebnisse zulässig sind.

Hinsichtlich der Optimalität der Informed Decomposition ist Folgendes festzustellen: Anhand der Endergebnisse wird ersichtlich, dass die Optimierungsergebnisse dieser Methode qualitativ unter denen der monolithischen Optimierung liegen – die optimale Masse ist bei Informed Decomposition durchschnittlich um 2 oder 3 größer. Allerdings liegt das Verhältnis von Massfehler zu Gesamtmasse bei unter zehn Prozent. Angesichts der Tatsache, dass die mit der in dieser Arbeit vorgestellten Methode erhaltenen Ergebnisse jedoch nur als Referenz für die tatsächliche Konstruktion verwendet werden, ist die Optimalität der erzielten Ergebnisse akzeptabel.

Darüber hinaus sind die Komplexität des Codes und die Rechenzeit zu berücksichtigen. Für die Optimierung eines einzelnen Systems wie des in dieser Arbeit behandelten, benötigt die monolithische Optimierung deutlich weniger Code und Rechenzeit. Ein monolithisches Optimierungsprogramm hat einen Code von etwa fünf- bis sechshundert Zeilen und seine Laufzeit während der Lösung eines dreidimensionalen Optimierungsproblems beträgt einige hundert Sekunden. Im Vergleich dazu besteht der Gesamtcode der Informed Decomposition aus mehr als 2000 Zeilen und die Lösung eines dreidimensionalen Optimierungsproblems nimmt ca. eine Woche in Anspruch. Der Vorteil der Informed Decomposition besteht jedoch darin, dass die Laufzeit des Codes hauptsächlich von Sampling und Metamodell-Training abhängt. Für eine bestimmte Struktur ist nur einmaliger Sampling- und Trainingsprozess erforderlich, zudem kann das entsprechende Metamodell direkt aufgerufen werden, wenn diese Struktur wieder verwendet wird. Deshalb liefert Informed Decomposition eine bessere Leistung, wenn das zu optimierende System eine komplexe Struktur besitzt.

Hinzu kommt, dass keiner der in dieser Arbeit ermittelten zulässigen Designräume eine ausreichend glatt Grenze hat, was darauf schließen lässt, dass der gesamte Designraum bis jetzt noch nicht vollständig gesampelt wird. Dies kann auch den Grund für die nicht genügende Optimalität von *Informed Decomposition* sein und könnte vielleicht mit präziseren Parametern und mehr generierten Samples verbessert werden.

## 8 Zusammenfassung und Ausblick

Ebenso wie die Arbeit von Kerscher (2021) zitiert diese Arbeit die moderne Strukturoptimierungsmethode Informed Decomposition (Krischer & Zimmermann, 2021) und die Modifikation active learning strategy (Krischer, 2022) für topologische Optimierung. Aber im Gegensatz zur Arbeit von Kerscher (2021), die darauf abzielt, die Topologie mit 2 Freiheitsgraden und 2 Interfaces zu optimieren, optimiert diese Arbeit Schritt für Schritt die ein-, zwei- und dreidimensionale Topologie mit 3 Freiheitsgrade und 2 Interfaces mit Hilfe von *Informed Decomposition*.

Im Laufe der Studie wurde der Einfluss diverser Parameter ermittelt, die daraufhin mit geeigneten Werten versehen wurden, um bessere Ergebnisse zu erzielen. Außerdem wurden Änderungen an der Struktur der Methode durchgeführt, sodass Informed Decomposition auch zur Optimierung von Topologien mit drei Freiheitsgraden und zwei Interfaces verwendet werden kann. Schließlich wurden die durch diese Methode erhaltenen Optimierungsergebnisse mit via monolithischer Optimierung gewonnenen standardisierten Ergebnissen verglichen, um die Anwendbarkeit und Optimalität der Informed Decomposition für das Optimierungsproblem der erwähnten Topologie zu bestimmen.

Bei der 1-D-Balken-Optimierung ist die optimierte Masse von *Informed Decomposition* um 3.36 größer als die optimierte Masse von monolithischer Optimierung, was einem prozentigen Fehler von nur 3.4% gegenüber der Gesamtmasse von 98.41 entspricht. Bei der 2D-Topologieoptimierung ist der prozentige Fehler von *Informed Decomposition* mit 28.33 % bei normaler Kraft am größten und mit 0.11 % bei der vollen Kraft am kleinsten. Bei der 3D-Topologieoptimierung beträgt der maximale Fehler 12.04 % bei Rotation und der minimale Fehler 0.10 % bei der vollen Kraft. Durch diesen Vergleich ließ sich bestätigen, dass Informed Decomposition eine geeignete Methode für die Optimierung dreidimensionaler Topologien mit drei Freiheitsgraden und zwei Interfaces darstellt.

Allerdings wurden in dieser Arbeit nur die Funktionen einiger Parameter diskutiert und nur ungefähre geeignete Werte gewählt. Eine feinere Abstimmung der Parameter könnte zu besseren Optimierungsergebnissen führen.

Ferner wurde nur die Optimierung eines Systems mit zwei Komponenten diskutiert. In zukünftigen Arbeiten könnte die hier vorgestellte Methode zur Optimierung komplexerer Systeme mit mehr Freiheitsgraden und Interfaces sowie von Systemen mit unterschiedlichen Materialien, Strukturen oder Verbindungsverhältnissen verwendet werden, um zu überprüfen, ob sich die Methode auch für weitere Topologie eignet. Durch die Optimierung komplexerer Strukturen wird auch eine stärkere Optimierung von Parametern und Code ermöglicht.

## 9 Literaturverzeichnis

- Alpaydin, E. (2010).** Introduction to Machine Learning. MIT Press
- 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. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-662-05086-6
- Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992).** A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), Proceedings of the fifth annual workshop on computational learning theory - colt '92 (pp. 144–152). New York, New York, USA: ACM Press. doi: 10.1145/130385.130401
- Cortes, C., & Vapnik, V. (1995).** Support-vector networks. *Machine Learning*, 20(3), 273–297. doi: 10.1007/BF00994018
- Ethem, A. (2020).** Introduction to Machine Learning (Fourth ed.). MIT. pp. xix, 1–3, 13–18.
- Guyan, R. J. (1965).** Reduction of stiffness and mass matrices. *AIAA Journal*, 3, 380.
- 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
- Hughes, T. J. (2000).** The finite element method: Linear static and dynamic finite element analysis (Repr. [der Ausg.] 1987, corr. minor errors ..., 1. publ ed.). Mineola, NY: Dover Publ.
- 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
- Kerscher, T. (2021).** Machine Learning Estimators for Structural Optimization. Masterarbeit, Lehrstuhl für Produktentwicklung und Leichtbau, Technische Universität München
- 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
- Krischer, L., Sureshbabu, A. V., & Zimmermann, M. (2020).** Modular topology optimization of a humanoid arm. In 2020 3rd international conference on control and robots (iccr). Tokyo, Japan: IEEE. doi: 10.1109/iccr51572.2020.9344316
- Krischer, L., & Zimmermann, M. (2021).** On the decomposition of structural optimization problems (submitted). *Structural and Multidisciplinary Optimization*.
- Krischer, Sureshbabu, Zimmermann, 2022.** Active-learning combined with topology optimization for Top-Down Design of multi-component systems.

- Liu, G. R., & Quek, S. S. (2013).** The finite element method: A practical course (2nd ed. ed.). Amsterdam: Elsevier/BH.
- 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
- Mitchell, T. M. (1997).** Machine Learning. New York: McGraw Hill.
- Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012).** Foundations of Machine Learning. The MIT Press
- O'Connell, R. F., Hassig, H. J., & Radovcich, N. A. (1976).** Derivatives of a statically reduced stiffness matrix with respect to sizing variables. *Journal of Aircraft*, 13(1), 59–60. doi: 10.2514/3.44510
- Petraroia, D. (2021).** Stiffness matrix for 8-node hexahedron. Retrieved 14.05.2021, from <https://de.mathworks.com/matlabcentral/fileexchange/67320-stiffness-matrix-for-8-node-hexahedron>.
- Russell, S. J., Norvig, P. (2010).** Artificial Intelligence: A Modern Approach (Third ed.). Prentice Hall.
- Rust, W. (2016).** Nichtlineare finite-elemente-berechnungen. Wiesbaden: Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-13378-8
- Schröder, D. (2010).** Intelligente Verfahren, *Identifikation und Regelung nichtlinearer Systeme*, Springer Heidelberg Dordrecht London New York, doi:10.1007/978-3-642-11398-7
- Sigmund, O. (1994).** Design of material structures using topology optimization (Dissertation). Technical University of Denmark, Lyngby, Denmark.
- Sigmund, O. (1997).** On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines*, 25(4), 493–524. doi: 10.1080/08905459708945415
- Sigmund, O. (2001).** A 99line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(2), 120–127. doi: 10.1007/s001580050176
- Sigmund, O., & Maute, K. (2013).** Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48(6), 1031–1055. doi: 10.1007/s00158-013-0978-6
- 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. (2002).** A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, 12(2), 555–573. doi: 10.1137/S1052623499362822
- The MathWorks Inc. (R2021b).** sparse: Create sparse matrix. Abgerufen am 07.01.2022, von <https://de.mathworks.com/help/matlab/ref/sparse.html>
- 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

**Zienkiewicz, O. C., Taylor, R. L., & Fox, D. A. (2013).** The finite element method for solid and structural mechanics (7th ed. ed.). Burlington: Elsevier Science.

**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: 10.1080/09544828.2017.130366

**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

## **Anhang**

<b>Anhang.....</b>	<b>A-1</b>
A1 2D Spannungselementsteifigkeitsmatrix .....	A-2
A2 $\kappa$ -Representation mit 6 Freiheitsgrad .....	A-3

## A1 2D Spannungselementsteifigkeitsmatrix

Bei der zweidimensionalen Finite-Elemente-Analyse vom Balken ist jede Finite-Elemente-Struktur ein kleines Quadrat, und ihre Länge und Breite sind wie folgt:

$$a = 0.49 \cdot \frac{l_x}{n_{ele,x}}; b = 0.49 \cdot \frac{l_y}{n_{ele,y}} \quad (A1 - 1)$$

Um eine Elementsteifigkeitsmatrix für lineare Spannungselemente in der Ebene mit vier Knoten zu bilden, werden einige Materialkonstanten, wie das Elastizitätsmodul  $E$  und die Poissonzahl  $\nu$ . Dann kann die Steifigkeitsmatrix des zweidimensionalen Finite-Elements geschrieben werden:

$$k_e = \frac{E \cdot h}{1 - \nu^2} \cdot \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 \\ k_2 & k_1 & k_8 & k_7 & k_6 & k_5 & k_4 & k_3 \\ k_3 & k_8 & k_1 & k_6 & k_7 & k_4 & k_5 & k_2 \\ k_4 & k_7 & k_6 & k_1 & k_8 & k_3 & k_2 & k_5 \\ k_5 & k_6 & k_7 & k_8 & k_1 & k_2 & k_3 & k_4 \\ k_6 & k_5 & k_4 & k_3 & k_2 & k_1 & k_8 & k_7 \\ k_7 & k_4 & k_5 & k_2 & k_3 & k_8 & k_1 & k_6 \\ k_8 & k_3 & k_2 & k_5 & k_4 & k_7 & k_6 & k_1 \end{bmatrix}$$
  

$$mit k = \begin{bmatrix} -\frac{\nu a^2 - 2b^2 - a^2}{6ab} \\ \frac{\nu}{8} + \frac{1}{8} \\ -\frac{\nu a^2 + 4b^2 - a^2}{12ab} \\ \frac{3\nu}{8} - \frac{1}{8} \\ \frac{\nu a^2 - 2b^2 - a^2}{12ab} \\ -\frac{\nu}{8} - \frac{1}{8} \\ \frac{\nu a^2 + b^2 - a^2}{6ab} \\ -\frac{3\nu}{8} + \frac{1}{8} \end{bmatrix} \quad (A1 - 2)$$

## A2 $\kappa$ -Representation mit 6 Freiheitsgrad

Die Gleichung (3-1) beschreibt die Steifigkeitsmatrix für einen Balken mit 6 Freiheitsgraden.

$$K = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad (A2-1)$$

Mit der gleichen Idee wie in Krischer and Zimmermann (2021) können 4 Parameter, nämlich der  $\kappa$ -Vektor mit 4 Elementen,

$$\kappa = [K_{11}, K_{22}, K_{33}, K_{66}] \quad (A2-2)$$

verwendet werden, um die  $6 \times 6$  Steifigkeitsmatrix für 6 Freiheitsgraden auszudrücken. Die originale Steifigkeitsmatrix  $K$  kann durch diese 4 Elemente beschrieben:

$$K = \begin{bmatrix} K_{11} & 0 & 0 & -K_{11} & 0 & 0 \\ 0 & K_{22} & \frac{K_{22} \cdot L^2 + K_{33} - K_{66}}{2L} & 0 & -K_{22} & \frac{K_{22} \cdot L^2 - K_{33} + K_{66}}{2L} \\ 0 & \frac{K_{22} \cdot L^2 + K_{33} - K_{66}}{2L} & K_{33} & 0 & -\frac{K_{22} \cdot L^2 + K_{33} - K_{66}}{2L} & \frac{K_{22} \cdot L^2 - K_{33} - K_{66}}{2} \\ -K_{11} & 0 & 0 & K_{11} & 0 & 0 \\ 0 & -K_{22} & -\frac{K_{22} \cdot L^2 + K_{33} - K_{66}}{2L} & 0 & K_{22} & -\frac{K_{22} \cdot L^2 - K_{33} + K_{66}}{2L} \\ 0 & \frac{K_{22} \cdot L^2 - K_{33} + K_{66}}{2L} & \frac{K_{22} \cdot L^2 - K_{33} - K_{66}}{2} & 0 & -\frac{K_{22} \cdot L^2 - K_{33} + K_{66}}{2L} & K_{66} \end{bmatrix} \quad (A2-3)$$