Diplomarbeit

Automatisierte Formoptimierung von Sandwichstrukturen

ausgeführt zum Zwecke der Erlangung des akademischen Grades einer Diplom-Ingenieurin (Dipl.-Ing. oder DI), eingereicht an der TU Wien, Fakultät für Maschinenwesen und Betriebswissenschaften, von

Sabrina ENGELHART

Mat.Nr.: 1326675

unter der Leitung von

Univ.Prof.in Dr.in-Ing.in Stefanie Elgeti

Ass.Prof.in Dipl.-Ing. Dr.in techn. Skrna-Jakl

Sebastian Hube M. Sc

Institut für Leichtbau und Struktur-Biomechanik

Nöstach, Oktober 2022

Ich nehme zur Kenntnis, dass ich zur Drucklegung dieser Arbeit nur mit Bewilligung der Prüfungskommission berechtigt bin.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde. Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.

Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis „Code of Conduct“ an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Stadt und Datum	*Unterschrift*

Kurzfassung

Bauteile möglichst leicht zu gestalten, ist gefragter denn je, denn durch die Verringerung der Masse von Bauteilen, können zeitgleich Energie und Ressourcen eingespart werden. Diese Gewichtsreduktion kann unteranderem durch den Einsatz von Optimierungsverfahren erreicht werden. In dieser Arbeit wird die automatische Formoptimierung von Bauteilen behandelt, genauer die automatische Optimierung der Geometrie eines Bauteiles. Durchgeführt wird diese automatische Formoptimierung mit der Optimierungssoftware Dakota und dem Finite-Elemente-Programm Abaqus.

Anhand von zwei Beispielen wird untersucht, ob und wie die automatische Formoptimierung mit Dakota und Abaqus umgesetzt werden kann. Beim ersten Berechnungsmodell handelt es sich um einen I-Träger und beim zweiten, um ein Sandwichbauteil. Bei beiden Beispielen soll durch das Variieren der Geometrie das Gewicht, die Spannungen und die Durchbiegungen eines Bauteils optimiert werden. Beim I-Träger wird die Formoptimierung, sowohl analytisch als auch numerisch mit Dakota und Abaqus durchgeführt. Durch den Vergleich der analytischen und numerischen Optimierungsergebnisse können die Lösungen überprüft werden.

Das Aufsuchen der analytischen Lösung für den I-Träger hat gezeigt, dass es sogar bei diesem einfachen Bauteil nur für besondere Einzelfälle eine analytische Optimierungslösung gibt. Bei Optimierungsaufgaben vom I-Träger mit nur einer Variable haben bei allen sechs verwendeten Optimierungsalgorithmen die numerischen Ergebnisse mit den analytischen übereingestimmt. Bei der I-Träger Optimierung mit zwei Variablen waren nur noch zwei Algorithmen erfolgreich. Mit zunehmender Variablenanzahl steigt also die Komplexität der Optimierungsaufgabe, und die größte Herausforderung ist die Wahl eines geeigneten Optimierungsalgorithmus. Das vorhandene Optimierungsprogramm kann mit geringem Aufwand für eine andere Geometrie adaptieren werden.

Abstract

This master thesis aims to explore an automatic shape optimization for structures, in other words the automatic optimization of the geometry from a component. It is more in demand than ever to make components as light as possible, because reducing the mass of components can save energy and resources at the same time. This weight reduction can be achieved, among other things, by using optimization methods.

In this work, the automatic shape optimization is carried out with the optimization software Dakota and the finite element program Abaqus. Two examples are used to examine whether and how automatic shape optimization can be implemented with Dakota and Abaqus. The first calculation model is an I-beam and the second is a sandwich component. In these examples, the weight, stresses and deflections of a component should be optimized by varying the geometry. For the I-beam, shape optimization is carried out both analytically and numerically with Dakota and Abaqus. The solutions can be checked by comparing the analytical and numerical optimization results.

The evaluation of the analytical solution for the I-beam has shown that even with this simple component there is only an analytical optimization solution for special individual cases. For I-beam optimization tasks with only one variable, the numerical results agreed with the analytical ones for all six employed optimization algorithms. Only two algorithms were successful in the I-beam optimization with two variables. With an increasing number of variables, the complexity of the optimization task increases and the greatest challenge is the choice of a suitable optimization algorithm. It is possible to adapt the existing optimization program for a different geometry with little effort.

Inhaltsverzeichnis

1 Motivation/Einleitung

Ziel des Leichtbaues ist es, Bauteile so zu gestalten, dass sie möglichst leicht sind und trotzdem den Anforderungen hinsichtlich ihres Einsatzes genügen. Gründe für die Anwendung von Leichtbau können unteranderem sein, dass er funktionsbedingt notwendig ist, um Kosten zu reduzieren oder um umweltfreundliche Bauteile herzustellen. Durch die Verringerung der Masse von Bauteilen, können zeitgleich Energie und Ressourcen eingespart werden. In Zeiten des Klimawandels und des Ressourcenschutzes besitzt der Leichtbau dahingehend ein hohes Potential. Neben, zum Beispiel der richtigen Materialauswahl, genaueren Berechnungsverfahren und konstruktiven Maßnahmen, kann Gewichtsreduktion auch durch den Einsatz von Optimierungsverfahren erreicht werden.

Unter den Begriff Optimierung versteht man im Allgemeinen die Suche nach der besten Lösung des zu untersuchenden Problems. Bei einer Formoptimierung wird bei dieser Suche die Geometrie des zu optimierenden Bauteiles variiert. Dies ist nur möglich, wenn ein Kriterium existiert, das ermöglicht eine Lösung als besser oder optimal einzustufen.

Neben einem iterativen, analytischen Vorgehen kann die Formoptimierung auch Computerbasiert mit einer Optimierungssoftware durchgeführt werden. Eine mögliche Software hierzu ist Dakota. Ist das Ziel der Optimierung eine verbesserte Form, spricht man von Formoptimierung. Dakota ermöglicht eine solche Formoptimierung, wobei anhand von Parametern die Geometrie variiert wird, um dem dazugehörigen Optimierungskriterium entsprechend, die beste Bauteilgeometrie zu ermitteln.

In dieser Arbeit ist das Ziel durch das Variieren der Geometrie die Spannungen und Durchbiegungen eines Bauteils zu optimieren. Dafür benötigt der Optimieralgorithmus von Dakota, die Information, wie sich diese Eigenschaften bei Variation der Geometrie verhalten. Zur Berechnung dieses Zusammenhangs kommt das Finite-Elemente-Programm Abaqus unter Anwendung eines parametergesteuerten Finite-Elemente-Modells des Bauteils zum Einsatz. Im Rahmen dieser Arbeit wird die Formoptimierungen mit der Optimierungssoftware Dakota und dem Finite-Elemente-Programm Abaqus durchgeführt.

Anhand von zwei Beispielen wird in dieser Arbeit untersucht, ob und wie die Formoptimierung mit Dakota und Abaqus umgesetzt werden kann. Beim ersten Berechnungsmodell handelt es sich um einen I-Träger, wobei das optimale Profil sowohl analytisch, als auch numerisch mit Dakota und Abaqus ermittelt wird. Durch den Vergleich der beiden Optimierungsergebnisse können die Lösungen überprüft werden. Bei dem zweiten Beispiel wird ein Sandwichkeil untersucht. Bei dem Sandwichkeil handelt es sich um ein Sandwichelement, bei welchen an einem Ende des Bauteiles die Deckschichten in einem Übergangsbereich zusammengeführt werden. In diesem Bereich nimmt die Höhe des Kerns kontinuierlich ab, bis sich die beiden Deckschichten berühren. Mit Hilfe von Dakota und Abaqus wird bei diesem Beispiel die optimale Form dieses Überganges ermittelt.

Diese Beispiele konnten verdeutlichen, dass nur für wenige, sehr wenige Spezialfälle eine analytische Formoptimierung möglich ist, und dass ein vorhandenes Optimierungsprogramm mit geringem Aufwand für eine andere Geometrie adaptieren werden kann.

2 Theoretischer Hintergrund

In diesem Kapitel wird der Begriff Optimierung und der mathematische Hintergrund dazu behandelt. Ferner werden auch die Optimierungsverfahren, welche in dieser Arbeit verwendet werden, vorgestellt. Da die hier vorgestellte Formoptimierung auf einer Finite-Elemente-Berechnung beruht, wird die Theorie dazu, hier ebenfalls behandelt. Ebenso wie der Aufbau eines Sandwichelements.

2.1 Optimierung

Unter dem Begriff Optimierung versteht man im Allgemeinen die Suche nach der besten Lösung des zu untersuchenden Problems. Dies ist nur möglich, wenn ein Bewertungskriterium existiert, das ermöglicht eine Lösung als besser oder optimal einzustufen.

2.1.1 Mathematische Grundlagen

Wenn man mit Hilfe von mathematischen Verfahren ein Optimum sucht, muss zuerst das betrachtete System mathematisch beschrieben werden, und daraus eine Zielfunktion als Bewertungskriterium abgeleitet werden. Bei der Formoptimierung eines Bauteils wird dieses anhand von unterschiedlichen Parametern, z.B. Länge, Breite beschrieben. Diese lassen sich in fix vorgegebene und jene, welche im Rahmen der Optimierung verändert werden können, unterteilen. Die Veränderlichen können verallgemeinert im n-dimensionalen Vektor $\underline{x} = (x_1, x_2, \dots, x_n)^T$ zusammengefast werden.

Definition: Optimierungsaufgabe mit Nebenbedingungen

Eine Optimierungsaufgabe mit Nebenbedingung besteht aus einer Zielfunktion $f(\underline{x})$, die unter der Betrachtung von Nebenbedingungen minimiert bzw. maximiert werden soll. Bei einem allgemeinen Problem betrachtet man also eine Zielfunktion f: $\mathbb{R}^n \to \mathbb{R}$ die einen Parametervektor $\underline{x} \in \mathbb{R}^n$ mit n Komponenten auf einen Skalar abbildet. Da in diesem Kapitel nur Optimierungsprobleme mit einer Zielfunktion behandelt werden, bildet f in die reellen Zahlen ab. Eine Minimierungsaufgabe lässt sich allgemein wie folgt definieren:

$$\begin{aligned} &\text{Minimiere} && f(\underline{x}) && \\ &\text{unter den Nebenbedingungen} && g_j(\underline{x}) = 0, && j = 1,2,\dots,m \\ & && h_i(\underline{x}) \le 0, && i = 1,2,\dots,l \\ & && x_k^{(U)} \le x_k \le x_k^{(O)} && k = 1,2,\dots,n \\ & && \underline{x} \in X\,. && \end{aligned} \tag{1}$$

Die Restriktionen $g_j : \mathbb{R}^n \to \mathbb{R}, j = 1,2,\dots,m$ beschreiben die Gleichheitsnebenbedingungen und $h_i : \mathbb{R}^n \to \mathbb{R}, i = 1,2,\dots,l$ die Ungleichheitsnebenbedingungen. Darüber hinaus können auch oberer Grenzen $x_k^{(O)}$ und untere Grenzen $x_k^{(U)}$ für einzelne Parameterwerte x_k

vorgegeben werden. Bei einer restringierten Optimierungsaufgabe ist der zulässige Bereich X die Menge aller $\underline{x} \in \mathbb{R}^n$, die innerhalb der Intervallgrenzen liegen und sowohl die Ungleichheit- als auch die Gleichheitsnebenbedingungen erfüllen.

Eine Nebenbedingung ist aktiv in $\underline{x} \in X$, wenn $h_i(\underline{x}) = 0$ und inaktiv, wenn $h_i(\underline{x}) < 0$ ist.

Bei den nachfolgenden Optimierungen handelt es sich stets um Minimierungsaufgaben. Das stellt aber keine Einschränkung dar, weil die Maximierung einer Funktion *f* übereinstimmt mit der Minimierung von *–f*.

Der Punkt $\underline{x}^*$ ist ein lokales Minimum der Funktion f, wenn alle zulässigen $\underline{x}$ in einer genügend kleinen Umgebung U rund um $\underline{x}^*$ keinen kleineren Funktionswert f haben, also gilt $f(\underline{x}^*) \leq f(\underline{x}) \; \forall \, \underline{x} \in U \subseteq X$.

Wenn es im gesamten zulässigen Bereich X keinen kleineren Funktionswert f gibt und daher $f(\underline{x}^*) \leq f(\underline{x}) \; \forall \underline{x} \in X$ erfüllt ist, besitzt f an der Stelle $\underline{x}^*$ ein globales Minimum.

Ist f_1 eine stetige Funktion und die zulässige Menge X nichtleer, abgeschlossen und beschränkt, dann existiert ein Punkt $\underline{x}^* \in X \subseteq \mathbb{R}^n$, bei welchen die Zielfunktion minimal ist. Sie besetzt also ein globales Minimum [2].

Es existiert leider kein direktes Kriterium, welches allgemein ein globales Minimum finden kann. Jedoch gibt es Optimalitätsbedingungen, welchen in lokalen Minima erfüllt sind. Desweiten gilt, dass jedes globale Minimum auch ein lokales Minimum ist. Man kann das globale Minimum also dadurch finden, dass man alle lokalen Minima innerhalb des zulässigen Bereichs ermittelt und anschließend ihrer Zielfunktionswerte vergleicht. Das lokale Minimum mit dem kleinsten Funktionswert ist schließlich das globale Minimum.

Optimalitätsbedingungen

Notwendige Optimalitätsbedingungen müssen in jedem lokalen Extrempunkt eines Optimierungsproblems erfüllt sein. Daher kann man mit ihnen mögliche Punkte für ein Minimum finden. Die hinreichenden Bedingungen helfen anschließend um zu entscheiden, ob es sich bei einen gefundenen Punkt tatsächlich um ein lokales Minimum handelt.

Optimalitätsbedingungen für Funktionen einer Variable ohne Nebenbedingungen

Die notwendige Bedingung 1. Ordnung für ein lokales Minimum einer differenzierbaren Funktion *f* mit einer Variable und ohne Nebenbedingungen ist $f'(x^*) = 0$. Diese Bedingung muss erfüllt sein, sonst kann x^* kein lokales Minimum sein.

Die hinreichende Bedingung lautet $f''(x^*) > 0$. Wenn diese Bedingung auch erfüllt ist, dann ist x^* ein lokales Minimum.

Die notwendige Bedingung 2. Ordnung besagt, dass nur, wenn $f''(x^*) \geq 0$, x^* ein lokales Minimum sein kann. Tritt der Fall ein, dass $f''(x^*) = 0$ ist, dann müssen höhere Ordnungen angeschaut werden. Allgemein gilt, die notwendige Bedingung, dass die niedrigste Ableitung, welche nicht Null ist, gerade sein muss. Dass diese positiv sein muss, ist eine notwendige und hinreichende Bedingung [2].

Mit diesen Bedingungen findet man nur ein Minimum, wenn die Zielfunktion nach unten beschränkt ist, sonst ist das Optimierungsproblem nicht lösbar [2].

Optimalitätsbedingungen für Funktionen in mehreren Variablen ohne Nebenbedingungen

Um einen möglichen Kandidat für ein lokales Minimum von einer Funktion mit n Variablen zu finden, muss folgende notwendige Bedingung 1. Ordnung erfüllt sein:

$$\frac{\partial f(\underline{x}^*)}{\partial x_k} = 0, \quad k = 1,2,\dots,n. \tag{2}$$

Außerdem verlangt die hinreichende Bedingung 2. Ordnung, dass die Hesse-Matrix $\mathbf{H}(\underline{x}^*)$ positiv definit ist. Die notwendige Bedingung 2. Ordnung für diesen Fall besagt, dass die Hesse-Matrix positiv semidefinit oder positiv definit sein muss.

$$\mathbf{H}(\mathbf{x}^*) = [\frac{\partial^2 f(\underline{x}^*)}{\partial x_k \partial x_j}]_{(nxn)} \quad k = 1,2,\dots,n \qquad j = 1,2,\dots,n \tag{3}$$

Optimalitätsbedingungen für Funktionen in mehreren Variablen mit Nebenbedingungen

Bei Optimierungsproblemen mit Nebenbedingungen gibt es prinzipiell zwei mögliche Arten von Lösungen. Eine Möglichkeit ist, dass das globale Minimum $\underline{x}^*$ im Inneren des zulässigen Bereichs X liegt. Die andere Möglichkeit ist, dass das globale Minimum $\underline{x}^*$ ein Randpunkt ist, $\underline{x}^*$ also auf den Rand der zulässigen Menge X liegt. Für ein Problem mit einer Variable können Vereinfachungen getroffen werden. Man kann das globale Minimum finden, indem man die Zielfunktionswerte von allen Punkten, welche innerhalb des zulässigen Bereiches liegen und die Optimalitätsbedingungen für eine Variable ohne Nebenbedingungen erfüllen, mit jenen Funktionswerten vom Rand des zulässigen Bereiches vergleichen. Jener von diesen Punkten, der den kleinsten Zielfunktionswert besitzt, ist das globale Minimum. Um den allgemeine Fall mit mehreren Variablen lösen zu können, wird die Lagrange-Funktion

$$L(\underline{x}, \underline{v}, \underline{u}, \underline{s}) = f(\underline{x}) + \sum_{j=1}^{m} v_j\, g_j(\underline{x}) + \sum_{i=1}^{l} u_i(h_i(\underline{x}) + s_i^2) \tag{4}$$

definiert. Für die Ungleichheitsbedingungen $h_i(\underline{x})$ wir die Schlupfvariable s_i^2 eingeführt. v_j und u_i sind die Lagrange Multiplikatoren. Die notwendige Bedingung 1. Ordnung besagt, dass, wenn alle Gradienten der aktiven Nebenbedingungen in $\underline{x}^*$ linear unabhängig sind und $\underline{x}^*$ ein lokales Minimum ist, dann gibt es Vektoren $\underline{u}^*$ und $\underline{v}^*$, so dass die 1. Ableitungen bezüglich s_i, u_i, v_j und x_k von L Null sind. Zusätzlich existiert auch noch die notwendige Bedingung, dass $u_i \geq 0$ sein muss[2].

Für ein konvexes Optimierungsproblem ist diese notwendige Bedingung auch hinreichend. Die weiteren Bedingungen für ein allgemeines Optimierungsproblem mit Nebenbedingungen ist in [2] angeführt. Ein Optimierungsproblem ist konvex, wenn sowohl die Zielfunktion als auch die Menge der zulässigen Punkte konvex ist. Eine Menge C ist konvex, wenn die

Verbindungsstecke zweier beliebiger Punkte der Menge C in der Menge liegen. Eine Funktion $f: \mathbb{R}^n \to \mathbb{R}$ ist genau dann konvex, wenn ihr Definitionsbereich eine konvexe Menge $A \subset \mathbb{R}^n$ ist und falls für alle $x, y \in A$ mit $x \neq y$ glit, das die Verbindungsstrecke zwischen (x, f(x)) und (y, f(y)) nie unterhalb des Graphen von f liegt [2].

2.1.2 Optimierungsverfahern

In diesem Kapitel werden die vier Optimierungsverfahren, welche in dieser Arbeit zur Anwendung kommen, klassifiziert und allgemein beschrieben. Wenn die Aufgabe darin besteht von einem gegebenen Punkt aus ein lokales Minimum zu finden, dann kann man einen lokalen Optimierer verwenden. Ist das Problem konvex, findet ein lokaler Algorithmus auch das globale Optimum. Eine weitere Möglichkeit, um mit einem lokalen Optimierer ein globales Optimum zu finden, ist, dass man unterschiedliche Startpunkte verwendet. Ferner gibt es auch globale Optimierer mit welchen man das absolute Minimum suchen kann. Meist stellt die Suche nach einem absoluten Extremwert eine größere Herausforderung dar, als die Suche nach einem lokalen, und die Lösbarkeit bei einer definierten maximalen Rechenzeit hängt stark von den Zielfunktionseigenschaften ab. In Dakota gibt es sieben globale Optimierungsmethoden, die alle gradientenfrei sind. Die Optimierer *coliny_ea*, *soga* und *moga,* zählen zu den evolutionären Algorithmen. *Ncsu_direct* und *coliny_direct,* implementieren das DIRECT Verfahren, und *efficient_global*, *surrogate_based_global* basieren auf der Efficient Global Optimization (EGO). Zusätzlich wurde auch noch der lokale Optimierer *coliny_cobyla* verwendet, welcher auf den COBYLA Algorithmus beruht [1].
Eine Möglichkeit der Kategorisierung von Optimierungsverfahren ist, diese in analytische und numerische Verfahren zu unterteilen. Bei den analytischen Verfahren wird versucht, die exakte Lösung durch das Lösen von Gleichungen zu ermitteln. Die analytische Ermittlung des Optimums erfolgt üblicher Weise mit den Optimalitätsbedingungen, für welche die Ableitungen der Zielfunktion benötigt werden. In Abschnitt 2.1.1 werden diese Bedingungen genauer beschrieben. Oft müssen jedoch numerische Verfahren verwendet werden, da die analytische Berechnung nicht möglich ist. Ein möglicher Grund dafür kann sein, dass die Zielfunktion nicht stetig differenzierbar ist oder, dass das Problem zu komplex ist. Numerische Optimierungsverfahren ermitteln die näherungsweise Lösung iterativ. Von der Art der Zielfunktion und der Effizienz des Verfahrens hängt es ab, wie schnell es konvergiert. In Abbildung 2.1 werden die numerischen Optimierungsmethoden, welche in dieser Arbeit verwendet werden, in deterministische und stochastische Verfahren unterteilt. Stochastische Verfahren beinhalten immer eine Zufallskomponente. Im Unterschied dazu, haben Deterministischen Verfahren keine Zufallskomponente, sie bestehen immer aus definierten und reproduzierbaren Zuständen.

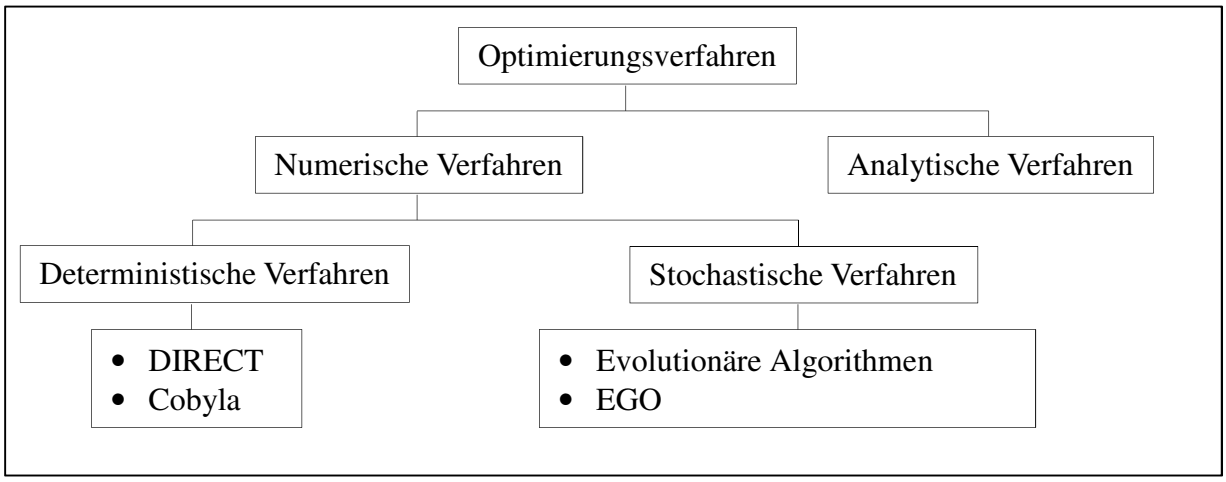


Abbildung 2.1: Klassifizierung der verwendeten Optimierungsverfahren

2.1.3 Deterministische Verfahren

Bei den deterministischen Verfahren gibt es keine Zufallskomponente, sie bestehen immer aus definierten und reproduzierbaren Zuständen. Das hat zur Folge, dass zu jedem Zeitpunkt des Verfahrens der nachfolgende Verarbeitungsschritt eindeutig festgelegt ist. Deshalb wird, wenn man bei einer Aufgabe bei mehrmaliger Anwendung immer die gleichen Startwerte verwendet, immer die gleiche Folge an Zwischenschritten durchlaufen und man bekommt die gleiche finale Lösung. Nur durch Variieren der Startwerte besteht die Möglichkeit, bei einem weiteren Durchlauf der Optimierung, eine bessere Lösung zu erhalten. Die Lösungen hängen stark von der Wahl der Startwerte ab. In der Regel konvergieren deterministische Optimierer sehr schnell, garantieren i.A. aber nur Konvergenz zu einen lokalen Optimum.

Dividing Rectangles Verfahren (DIRECT)

Das DIRECT Verfahren ist eine globale, deterministische, gradientenfreie Optimierungsmethode. Sie findet meistens rasch den Bereich, in welchem das Optimum liegt, jedoch dauert es dann meistens recht lange bis sie das genaue Optimum gefunden hat. Es hat sich daher herausgestellt, dass es sich anbietet, den Bereich für das Optimum mit den globalen DIRECT Verfahren zu bestimmen und dann mit einen lokalem Optimierer das genaue Optimum zu suchen [5].

Dieses Verfahren beruht darauf, dass der zulässige Bereich iterativ immer wieder in kleinere Teilintervalle zerlegt wird und die Zielfunktion im Mittelpunkt c des Intervalls ausgewertet wird. Als ersten Schritt wird der zulässige Bereich entsprechend der Anzahl der Variablen auf einen Hyper-Würfel normiert und im Mittelpunkt die Zielfunktion $f(c)$ ausgewertet. Damit die Mittelpunkte der bereits vorhandenen Intervalle auch nach der Teilung wieder Mittelpunkte der neuen Teilintervalle sind, wird ein vorhandenes Intervall nicht in zwei, sondern in drei gleiche Teilintervalle unterteilt. Dadurch, dass immer entlang der längeren Seite geteilt wird, entstehen keine länglichen Hyperrechtecke und die Suche ist sehr effektiv. Wie in Abbildung 2.2 für den zweidimensionalen Fall dargestellt, ist d der größte Abstand

vom Mittelunkt eines Hyperrechtecks zu seiner Ecke. Von den m Hyperrechtecken wird immer das potentiell optimale Rechteck j geteilt, für welches gilt:

$$f(c_j) - K\,d_j \leq f(c_i) - K\,d_i, \qquad i = 1, \ldots, m \text{ [6]} \tag{5}$$

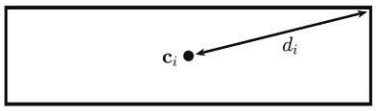


Abbildung 2.2: Hyperrechteck [6]

Der Wert der Konstante K beeinflusst ob mehr lokal in vielversprechenden Bereichen oder mehr global in noch nicht erforschten Bereichen gesucht wird. Im Fall von zwei Variablen ist der Hyperwürfel zweidimensional und Abbildung 2.3 zeigt eine mögliche Teilung. Die potentiell optimalen Rechtecke sind grün unterlegt und die neu generierten Mittelpunkte sind rot eingefärbt.

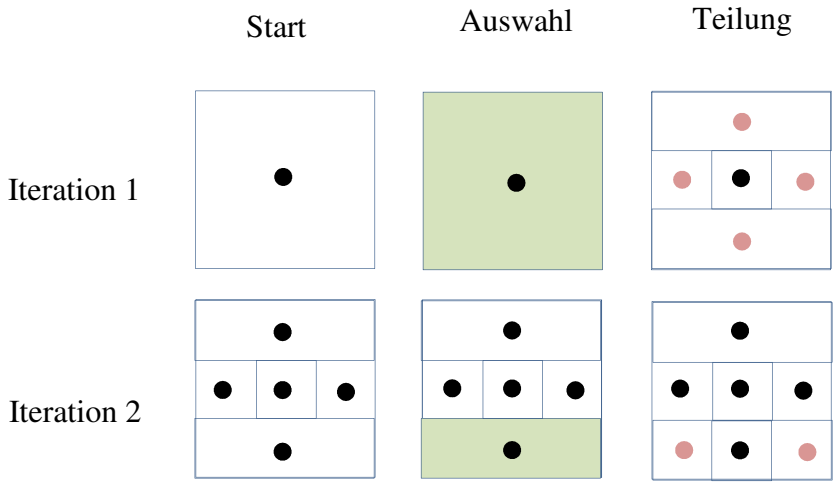


Abbildung 2.3: Mögliche Iterationen des DIRECT Algorithmus. Modifiziert nach [6]

In Dakota gibt es die zwei Implementierungen *ncsu_direct* und *coliny_direct*, welche auf dem DIRECT Verfahren beruhen. *Ncsu_direct* kann für die Optimierung verwendet werden, hat jedoch den Nachteil, dass es keine linearen und nichtlinearen Nebenbedingungen unterstützt. Es können also für die Spannung und für die Durchbiegung keine Grenzen vorgegeben werden.

COBYLA

Die Abkürzung COBYLA steht für *Constrained Optimization BY Linear Approximations* und ist eine Erweiterung des Nelder-Mead-Simplex-Algorithmus. Dieser basiert auf der Arbeit von Powell [10]. Von Powell selbst, wird COBYLA nur für eine geringe Anzahl an Optimierungsvariablen empfohlen. Es handelt sich hierbei um einen lokalen Optimierer.

Dieser Algorithmus approximiert die Zielfunktion und die Nebenbedingungen durch lineare Funktionen. Für ein Problem mit n Optimierungsvariablen wird mit einer geometrischen Figur mit n+1 Ecken gebildet. Diese geometrische Figur ergibt bei 2 Variablen ein Dreieck, bei 3 ein Tetraeder oder allgemein ein reguläres Polyeder, welches Simplex genannt wird. An den

Ecken des Simplex werden die Zielfunktionen und die Nebenbedingungen ausgewertet. Die Approximation wird anschließend unter der Voraussetzung, dass sie genau durch diese n+1 Stützstellen verlaufen muss, erzeugt. Die Funktionswerte der Approximation an den Stützstellen entsprechen also den Funktionswerten der approximierten Funktion. In jedem Iterationsschritt wird ein bestehender Eckpunkt durch einen neuen ersetzt. Dafür muss zuerst der aktuell beste Eckpunkt ermittelt werden. Dies geschieht mithilfe der Gütefunktion. Diese berücksichtigt nicht nur den Wert der Zielfunktion, sondern auch wie stark die Nebenbedingungen verletzt werden. Dann wird das linearisierte Optimierungsproblem, welches sich aus der linearen Zielfunktion, den linearen Nebenbedingungen und der Bedingung, dass der neue Punkt innerhalb einer Kugel mit den Radius ρ um die beste bisher gefundene Ecke liegen muss, gelöst, um den neuen Eckpunkt zu erhalten. Anschließend wird überprüft, ob sich die Gütefunktion der tatsächlichen Funktion stärker verbessert hat, als die Gütefunktion der linearen Näherung. Ist dies nicht der Fall, wird ρ verkleinert. Ebenfalls verkleinert wird dieser Wert, wenn ein optimaler Eckpunkt mit einem Abstand deutlich kleiner als ρ gefunden wird. ρ wird solange monoton verkleinert, bis ein vordefinierter Wert unterschritten wird. Dann ist die Optimierung beendet.
In Dakota gibt es den lokalen Optimierer *coliny_cobyla*, welcher auf dem COBYLA Optimierungsalgorithmus beruht.

2.1.4 Stochastische Verfahren

In dieser Arbeit werden zwei stochastische Verfahren verwendet: Evolutionäre Algorithmen und die Efficient Global Optimization. Stochastische Verfahren beinhalten immer eine Zufallskomponente. Dadurch besitzen sie die Fähigkeit, Suchbereiche mit lokalen Extremstellen zufällig verlassen zu können, um in anderen Bereichen nach dem Optimum weiter zu suchen. Ein lokales Optimum kann also überwunden werden, und sie sind daher geeignet, um globale Extremwerte zu finden.

Evolutionäre Algorithmen

Evolutionäre Algorithmen ahmen das Prinzip des Überlebens des Stärkeren bzw. des Anpassungsfähigsten und das Prinzip des zufälligen Informationsaustausches nach. Diese Prinzipien sind angelehnt an die biologische Evolution. Außerdem werden aus der Biologie stammende Konzepte, wie die Selektion, die Mutation und die Rekombination verwendet.
In Abbildung 2.4 ist der Aufbau eines evolutionären Algorithmus als Flussdiagramm dargestellt. Zu Beginn der Optimierung werden zufällige Punkte innerhalb des zulässigen Bereichs ausgewählt, diese stellen die Ausgangspopulation dar. Es wir an diesen Punkten die Zielfunktion ausgewertet, um anhand einer vordefinierten Bewertungsfunktion diese Punkte zu analysiert und zu sortiert. Durch Kreuzungen und Mutationen innerhalb der Ausgangspopulation werden anschließend neue Auswertungspunkte ausgewählt. Diese Punkte stellen dann eine neue Population dar und es werden die Zielfunktionswerte der neuen Population berechnet. Diese neue Population enthält außerdem auch noch den Punkt mit dem besten Zielfunktionswert der vorherigen Population. Erfüllt die neue Population das

Abbruchkriterium der Optimierung nicht, wird die neue Population zur Ausgangspopulation und es wird nach dem gleichen Prinzip, wie zuvor eine neue Population erzeugt. Dieser Vorgang wird so lange wiederholt, bis das Abbruchkriterium erfüllt ist [9].

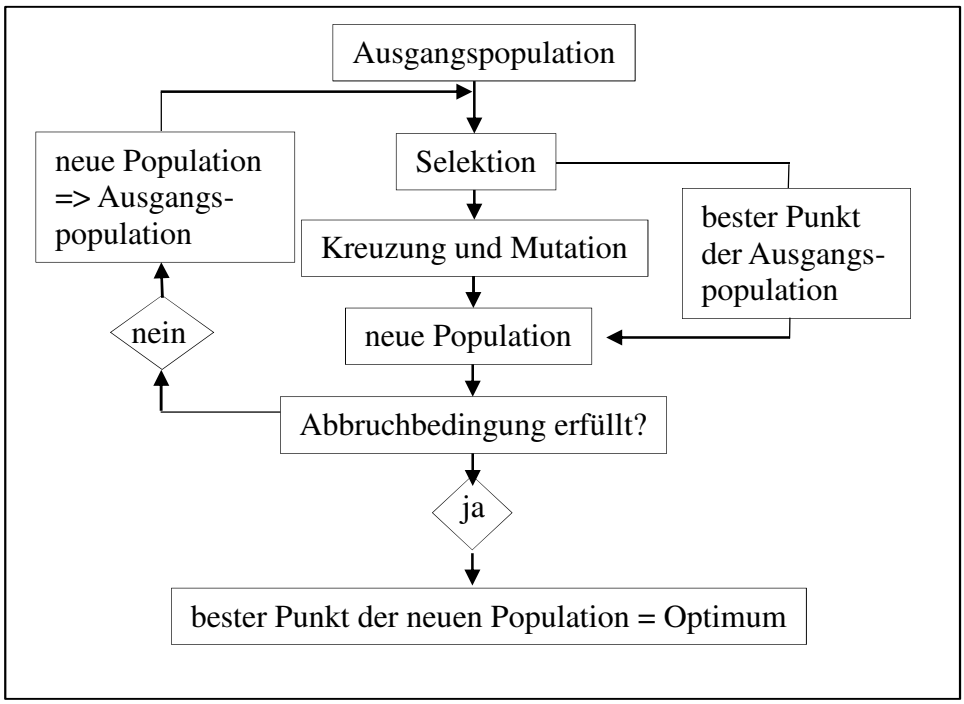


Abbildung 2.4: Aufbau eines evolutionären Algorithmus

Dakota beinhaltet drei evolutionäre Algorithmen, *coliny_ea*, *soga* und *moga*, wobei *moga* für multi-objektive-Probleme, also Probleme mit mehreren Zielfunktionen, verwendet wird und hier nicht zur Anwendung kommt.

Efficient Global Optimization (EGO)

Dieses Verfahren eignet sich besonders für Aufgaben mit zeitaufwendiger Zielfunktionsauswertung, da ein Ersatzmodell verwendet wird, bei welchem der Zielbereich so vereinfacht ist, dass eine schnellere Funktionsauswertung möglich ist. Weiters muss das Ersatzmodell so aufgebaut sein, dass es den Wertebereich gut approximieren kann.
Zu Beginn wird die Zielfunktion an zufällig ausgewählten Punkten ausgewertet und daraus das Ersatzmodell erzeugt. Mit Hilfe des Ersatzmodells werden jetzt neue erfolgversprechende Punkte ermittelt, welche das Modell verbessern. Diese Suche stellt eine eigene Optimierungsaufgabe dar, welche mit einem beliebigen Verfahren gelöst werden kann. Gesucht werden entweder Punkte, welche in einem Gebiet liegen, in welchen das Ersatzmodell eine hohe Unsicherheit besitzt, man also nicht weiß, ob es das tatsächliche Modell gut approximiert, oder Punkte, an welchen ein Optimum vermutet wird. An diesen Punkten wird anschließend die Zielfunktion ausgewertet und durch die erlangte Information das Ersatzmodell verbessert. Das Modell wird also schrittweise verbessert und das globale Optimum immer besser approximiert. Es werden so lange neue Punkte gesucht, bis ein Abbruchkriterium erfüllt wird [7, 8].

Die Optimierer *efficient_global* und *surrogate_based_global* von Dakota beruhen auf der Efficient Global Optimization.

2.2 Finite-Elemente-Methode

Die Finite-Elemente-Methode ist ein computergestütztes numerisches Berechnungsverfahren zur Berechnung von physikalischen und ingenieurwissenschaftlichen Problemen. Es gibt viele unterschiedliche Anwendungen, zum Beispiel kann sie für die Festigkeit- und Verformungsuntersuchungen von Bauteilen verwendet werden. Das zu untersuchende Bauteil wird für die Berechnung in kleine Elemente mit einer einfachen Geometrie unterteilt, den Finite-Elementen. Wenn es sich um Elemente mit linearen Ansatzfunktionen zur Näherung des Verschiebungsfeldes handelt, befindet sich an jeder Ecke des Elements ein Knoten. Bei Elementen mit quadratischen Ansatzfunktionen zur Näherung des Verschiebungsfeldes, gibt es zusätzlich zu den Eckknoten auch noch Knoten in der Mitte jeder Kante. Damit wird das Verhalten der Verschiebungen auf Elementebene näherungsweise formuliert, um dann das Verhalten der Gesamtstruktur aus der Kombination des Verhaltens der einzelnen Elemente zu ermittelten. Üblich ist, dass bei Festigkeits- und Verformungsuntersuchungen die Verschiebungen der Knotenpunkte in den Koordinatenachsen primär die Unbekannten sind, und das Verschiebefeld durch spezielle Ritz-Ansätze approximiert wird. Die Verschiebungen werden daher nur in den Knoten berechnet, in jeden andern Punkt des Elements werden sie interpoliert. Wenn einmal das Verschiebefeld des Körpers berechnet ist, können daraus die Verzerrungen und dadurch auch die Spannungen berechnet werden. Diese Vorgehensweise wird verschiebungsorientierte Finite-Elemente-Methode genannt und die vorliegende Arbeit basiert darauf. Da in dieser Arbeit die lineare Finit-Elemente-Methode verwendet wird, beschränkt sich die Erklärung auf diese.
Eine Finite-Elemente-Berechnung kann in drei Abschnitte unterteilt werden, die Modellbildung, die Modellberechnung und die Ergebnisbeurteilung. Zu Beginn muss definiert werden, welche Fragestellungen gelöst werden sollen, welche Näherungen getroffen werden können und wie das Modell dafür gewählt werden muss. Danach kann mit der Modellerstellung begonnen werden. Das geschieht computerbasiert mit dem Präprozessor eines Finite-Elemente-Programms. Dabei werden die Netzgeometrie, die Randbedingungen, die Art der Elemente und deren Materialeigenschaften festgelegt. Nach der Erstellung des Modells kann die Modellberechnung, mit dem Finte-Elemente-Programm durchführt werden. Falls diese erfolgreich abgeschlossen wird, kann das Ergebnis graphisch dargestellt und die Ergebnisse beurteilt werden.

2.2.1 Linearisierte Elastizitätstheorie

Um die Theorie zu der linearen Finiten-Elemente-Methode beschreiben zu können, werden einige Grundlagen benötigt. Wie der Name schon sagt, beruht diese Methode auf der linearisierten Elastizitätstheorie. Daher wird die linearisierte Verschiebungs-Verzerrungs-Bedingung

$$\tilde{\varepsilon} = \underline{\underline{d}}\,\underline{u} \tag{6}$$

verwendet. Für den dreiachsigen Verzerrungszustand sieht diese Beziehung, wenn sie in kartesischen Koordinaten dargestellt wird, folgendermaßen aus:

$$\begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yx} \\ \gamma_{zx} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{7}$$

Der Vektor der Verzerrungskomponenten $\tilde{\varepsilon}$ ist nur ein mathematischer Vektor, der die Komponenten des Verzerrungstensors enthält, $\underline{u}$ ist der Verschiebungsvektor und $\underline{\underline{d}}$ wird Differentialoperatormatrix genannt. Um das Materialverhalten berücksichtigen zu können, wird ein Materialgesetz benötigt, welches einen Zusammenhang zwischen den Spannungs- und Verformungszustand in einen Körper darstellt.

Analog zum Vektor der Verzerrungskomponenten wird auch der Vektor der Spannungskomponenten als mathematischer Vektor $\tilde{\sigma}$ definiert und $\underline{\underline{E}}$ ist der Elastizitätstensor.

$$\tilde{\sigma} = \underline{\underline{E}}\,\tilde{\varepsilon} \tag{8}$$

Für einen isotropen, linear elastischen Werkstoff gibt es nur zwei unabhängige Materialparameter, den E-Modul E und die Poissonzahl ν. Wenn das Materialgesetz für diesen Werkstoff in kartesischen Koordinaten dargestellt wird, nimmt es folgende Form an:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{yx} \\ \sigma_{zx} \end{pmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & & & \\ \nu & 1-\nu & \nu & & \underline{\underline{0}} & \\ \nu & \nu & 1-\nu & & & \\ & & & \frac{1-2\nu}{2} & 0 & 0 \\ & \underline{\underline{0}} & & 0 & \frac{1-2\nu}{2} & 0 \\ & & & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yx} \\ \gamma_{zx} \end{pmatrix}. \tag{9}$$

2.2.2 Spezieller Ritz´scher Ansatz

Bei der verschiebungsorientierten Finite-Elemente-Methode wird der Verschiebevektor $\underline{u}$ an einer beliebigen Stelle (x,y,z) im Bauteil mit Hilfe eines speziellen Ritz´schen Ansatz dargestellt. Die Herleitung, der in diesen Abschnitt angeführten Formel und weiterführende Informationen, können in [12] nachgelesen werden. Ein allgemeiner Ansatz lautet wie folgt:

$$\underline{u}(x,y,z) = \sum_i \varphi_{ui}(x,y,z)\, a_{ui} \,. \qquad (10)$$

Er besteht aus den Interpolationsfunktion φ_{ui} und den dazugehörigen Koeffizienten a_{ui}. Für die Finite-Elemente-Methode wird die Interpolationsfunktion so definiert, dass die dazugehörigen Koeffizienten, die Verschiebungen $\underline{U}$ diskreter, fest vorgegebener Punkte im Bauteil sind.

Weiters wird die Interpolationsfunktion nicht über das ganze Bauteil definiert, sondern nur für endlich groß Teilbereiche, welche auch Finite-Elemente genannt werden. Diese Teilgebiete sollen eine möglichst einfache Geometrie aufweisen, beispielsweise ein Rechteck. Das ganze Bauteil wird also in kleine Teilbereiche / finite Elemente unterteilt, wobei ein Finite-Elemente-Netz entsteht, siehe Abbildung 2.5.

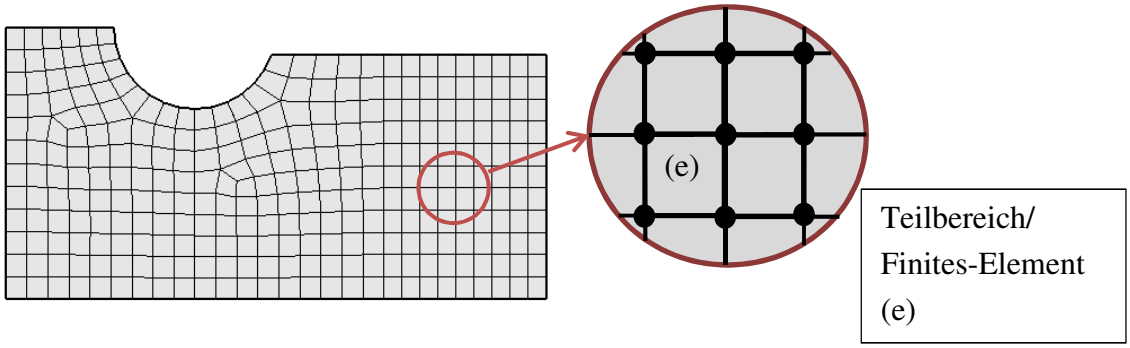


Abbildung 2.5: Finte-Elemente-Netz

Jedem finiten Element sind eine bestimmte Anzahl der zuvor erwähnten diskreten, fest vorgegebenen Punkte zugeordnet, welche Knoten genannt werden. Diese können unter anderem an den Ecken der Elemente positioniert sein. Die Verschiebungen der Knoten von einem Element in die jeweiligen Koordinatenrichtungen, werden in dem mathematischen Vektor $\tilde{U}^{(e)}$ zusammengefasst. Der Ansatz, welcher sich auf ein Element beschränkt, hat folgende Form:

$$\underline{u}(x,y,z) = \underline{\underline{\varphi}}_{ui}^{(e)}(x,y,z)\ \tilde{U}^{(e)} \,. \qquad (11)$$

Wenn die Knotenpunktverschiebungen $\tilde{U}^{(e)}$ eines Elements bekannt sind, kann der Verschiebevektor $\underline{u}(x,y,z)$ jedes beliebigen Punktes des Teilgebietes ermittelt werden. Die gewählten Interpolationsfunktionen müssen die kinematischen Randbedingungen und die Verträglichkeitsbedingungen erfüllen. Bei gleichartigen Elementen können gleichartige Interpolationsfunktionen verwendet werden.

Anhand dieses Ansatzes und mit dem Prinzip der virtuellen Arbeit kann die Elementsteifigkeitsmatrix $\underline{\underline{K}}^{(e)}$ hergeleitet werden, siehe Quelle [12]. Auf das globale Koordinatensystem bezogen, ist sie folgendermaßen definiert:

$$\underline{\underline{K}}^{(e)} = \int_{Vol^{(e)}} \underline{\underline{D}}^{(e)^T}\, \underline{\underline{E}}\, \underline{\underline{D}}^{(e)} dVol \,. \qquad (12)$$

Die D-Matrix $\underline{\underline{D}}^{(e)}$ beinhaltet die Interpolationsfunktionen des Elements und die Differentialoperatormatrix $\underline{\underline{d}}$, welche gemeinsam mit den Elastizitätstensor $\underline{\underline{E}}$ schon im Rahmen der linearisierten Elastizitätstheorie erwähnt wurden:

$$\underline{\underline{D}}^{(e)} = \underline{\underline{d}}\,\underline{\underline{\varphi}}^{(e)^T}(x,y,z)\,. \tag{13}$$

Es lässt sich folgendes Gleichungssystem für ein Element aufschreiben:

$$\underline{\underline{K}}^{(e)}\,\tilde{U}^{(e)} = \tilde{F}^{(e)}_{ges}\,. \tag{14}$$

Der Elementlastvektor $\tilde{F}^{(e)}_{ges}$ beinhaltet alle Kräfte, welche auf das Element wirken, also auch die unbekannten Knotenkräfte. Das sind jene Schnittkräfte, welche durch das gedankliche Herausschneiden des Elementes aus dem Bauteil entstehen. Aufgrund dieser unbekannten Knotenkräfte kann dieses Gleichungssystem noch nicht nach den Knotenpunktverschiebungen $\tilde{U}^{(e)}$ aufgelöst werden. Es ist daher die Assemblierung eines Gesamtsystems erforderlich. Dafür muss die Gesamtsteifigkeitsmatrix $\underline{\underline{K}}$ aus den Elementsteifigkeitsmatrizen des Bauteiles gebildet werden, sowie der Gesamtlastvektor $\tilde{F}$ aus den Elementlastvektoren. Die Schnittkräfte heben sich durch diese Assemblierung auf, und der Vektor beinhaltet schließlich nur noch die bekannten äußeren Lasten. Der Gesamtlastvektor $\tilde{F}$ setzt sich zusammen aus den Einzelkräften $\tilde{F}_C$, den Volumskräften $\tilde{F}_V$, den Oberflächenkräften $\tilde{F}_O$ und den Eigenspannungen $\tilde{F}_E$.

$$\tilde{F} = \tilde{F}_C + \tilde{F}_V + \tilde{F}_O + \tilde{F}_E \tag{15}$$

Da der Gesamtlastvektor $\tilde{F}$ aus den Element-Lastvektoren gebildet wird, müssen diese zuerst aus den verteilten Lasten, also der Volumenkraftdichte $\underline{q_v}$, der Oberflächenkraftdichte $\underline{q_o}$ und den Eigenspannungen $\tilde{\sigma}_E$ ermittelt werden. Das geschieht mit folgenden Formeln:

$$\tilde{F}^{(e)}_V = \int_{Vol^{(e)}} \underline{\underline{\varphi}}^{(e)}(x,y,z)\;\underline{q_v}(x,y,z)dVol \tag{16}$$

$$\tilde{F}^{(e)}_O = \int_{O^{(e)}} \underline{\underline{\varphi}}^{(e)}(x_O,y_O,z_O)\;\underline{q_o}(x_O,y_O,z_O)dO$$

$$\tilde{F}^{(e)}_E = -\int_{Vol^{(e)}} \underline{\underline{D}}^{(e)^T}(x,y,z)\;\tilde{\sigma}_E(x,y,z)dVol\,.$$

Nach der Assemblierung kann das Gleichungssystem

$$\underline{\underline{K}}\;\tilde{U} = \tilde{F} \tag{17}$$

nach dem globalen Verschiebungsvektor $\tilde{U}$, welcher die Knotenpunktverschiebungen beinhaltet, aufgelöst werden. Aufgrund von dem Zusammenhang $\underline{u}(x,y,z) = \underline{\underline{\varphi}}^{(e)}_{ui}(x,y,z)\;\tilde{U}^{(e)}$ ist der Verschiebungszustand des ganzen Bauteils bekannt.

Der Vektor der Verzerrungskomponenten kann mit folgender Formel berechnet werden:

$$\tilde{\varepsilon}^{(e)}(x,y,z) = \underline{\underline{D}}^{(e)}(x,y,z)\ \tilde{U}^{(e)} \tag{18}$$

und die Spannungen mit

$$\tilde{\sigma}_E{}^{(e)}(x,y,z) = \underline{\underline{E}}^{(e)}\tilde{\varepsilon}^{(e)}(x,y,z)\,. \tag{19}$$

Verschiebungen sind an den Elementgrenzen stetig, Spannungen jedoch nicht. Die bis jetzt erwähnen Integrale (12) und (16) der einzelnen Elemente werden numerisch berechnet. Die meisten Finite-Elemente-Programme verwenden dafür die Gauß´sche Integration.

$$\int_{Vol} \psi(x,y,z)dVol \approx \sum_{r=1}^{mxnxo} \psi(x,y,z)_r\ w_r \tag{20}$$

Bei der Gauß´schen Integration wird der Integrand $\psi(x,y,z)$ an bestimmten Integrationspunkten/Stützstellen $(x,y,z)_r$ ausgewertet und mit den dazugehörigen Integrationsgewichten w_r multipliziert. $mxnxo$ ist die Integrationsordnung, die angibt wie viele Stützstellen es gibt. Das Auffinden der Integrationsstützstellen und das Bestimmen der Gewichte ist sehr aufwendig und muss für jedes Element individuell durchgeführt werden. Isoparametrische Elemente umgehen dies, indem bei diesen Elementen das Element vom Originalraum auf ein Einheitselement im Bildraum transformiert wird. Der Integrand ist dadurch eine Funktion des Bildraumes, und er wird im diesem integriert. Da der Bildbereich für gleichartige Elemente ident ist, ergibt sich daraus der Vorteil, dass bei ihnen auch die Integrationsstützstellen und die dazugehörigen Gewichte gleich sind. Allerdings muss jetzt die Transformation ausgewertet werden.

2.2.3 Typische Finite Elemente

Es gibt zwei unterschiedliche Arten von Finiten-Elementen, Kontinuums- und Strukturelemente. Im Unterschied zu Kontinuumselementen, bei welchen das Verschiebungsfeld über die Interpolationsfunktionen beschrieben wird, wird bei Strukturelementen bereits ein Teil des Verschiebungsfeldes durch kinematische Bedingungen festgelegt. Typische Strukturelemente sind Schalen- und Balkenelemente. Welche Verschiebungsfreiheitsgrade die Knoten eines Elementes haben, hängt von der genauen Art des Elementes ab. Kontinuumselemente besitzen zum Beispiel nur translatorische Freiheitsgrade, Balkenelemente und Schalenelemente dagegen haben translatorische und rotatorische Freiheitsgrade.

2.3 Sandwichbauteile

In der Regel besteht ein Sandwichelement aus zwei dünnen Deckschichten und einem dicken Kern, wie in Abbildung 2.6. dargestellt. Ziel dieses Aufbaues ist es, dass ein Sandwichbauteil hohe mechanische Festigkeit und Steifigkeit bei einem geringen Gewicht aufweist. Sie eigenen sich besonders für die Übertragung von Membran- und Biegebeanspruchungen.

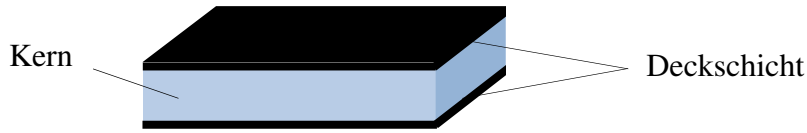


Abbildung 2.6: Aufbau eines Sandwichelements

Die Deckschichten sollen hochbeanspruchbar und dehnsteif sein, da sie die aus äußeren Belastungen entstehenden, tangential zur Bauteiloberfläche verlaufenden Zug – und Druckkräfte aufnehmen. Der Kern hat die Aufgabe, die Querkräfte aufzunehmen, die Deckschichten abzustützen und auf Distanz zu halten. Er soll daher leicht und ausreichend schubsteif sein. Je größer der Abstand zwischen den Decksichten ist, umso größer ist das Flächenträgheitsmoment des Querschnittes und damit auch die Biegesteifigkeit. Typische Werkstoffe für die Deckschichten sind Metalle und faserverstärkte Kunststoffe. Als Kernmaterial können Metallwaben, harzgetränkte Papierwaben, Hartschäume oder andere leichte Werkstoffe verwendet werden. Sehr wichtig ist, dass die Verbindung zwischen den Deckschichten und den Kern möglichst zug- und auch schubfest ist. Versagt diese Verbindung, verringert sich die Steifigkeit und Festigkeit des Bauteiles drastisch [10].

Aufgrund der geringen Dichte hat das Kernmaterial auch geringe Festigkeitswerte. Daher reagiert ein Sandwichelement empfindlich auf lokal eingeleitete Belastungen, weswegen an den Krafteinleitungsstellen zusätzliche konstruktive Maßnahmen erforderlich sind. Das Ziel dieser Maßnahmen ist, Spannungsspitzen zu mindern, indem die Krafteinleitungsfläche vergrößert wird. Meist wird der Kernwerkstoff durch ein steiferes oder festeres Material ersetzt oder die Deckschichten werden gezielt zusammengeführt. In Abbildung 2.7 sind drei Gestaltungskonzepte für die Krafteinleitung in Sandwichbauteilen dargestellt. Die Variante a) zeigt einen Onset, b) das Zusammenführen der Deckschichten und c) einen Insert. Um das Sandwichbauteil an den Kanten zu schützen und bei Bedarf auch Belastungen einleiten zu können, sind an den Enden und Ecken von Sandwichbauteilen oft zusätzliche Versteifungen notwendig, siehe in Abbildung 2.8. Kommt ein Kernmaterial in Wabenbauweise oder ein offenporiger Schaum zum Einsatz muss die Kantenversteifung wasserdicht ausgeführt sein um das Eindringen von Wasser zu verhindern.

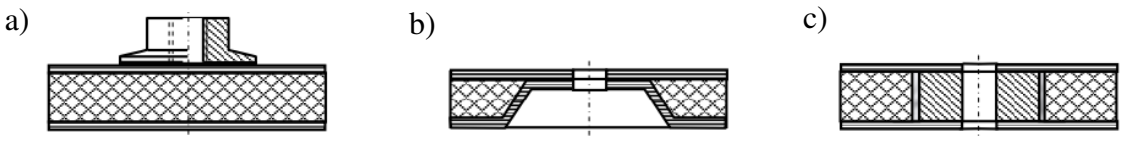


Abbildung 2.7: Gestaltungskonzepte von Sandwich-Krafteinleitungen [11]
a) Onset, b) Zusammenführung der Deckschichten, c) einen Insert

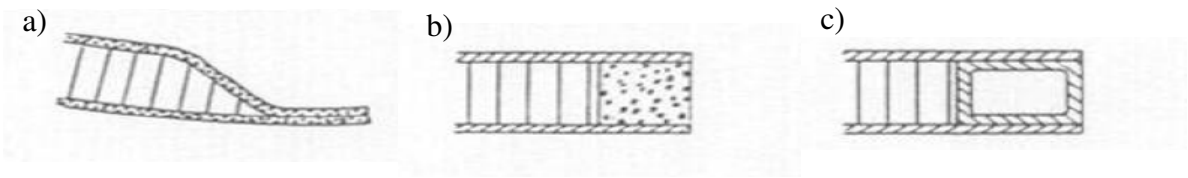


Abbildung 2.8: Kantenversteifungen von Sandwichbauteilen [12]
a) Zusammenführung der Deckschichten, b) Vollmaterial, c) Profil

3 Methoden

Wie schon Anfangs erwähnt, wird bei einer Formoptimierung, durch das Variieren der Geometrie eines Bauteiles, versucht vorab definierte Eigenschaften des Bauteiles zu optimieren. Um dies zu realisieren, müssen Bewertungskriterien existieren, welche es ermöglichen, die von der Geometrie abhängigen Werte von den Eigenschaften als besser einzustufen. Da mit Hilfe von mathematischen Verfahren nach dem Optimum gesucht wird, wird das betrachtende System mathematisch beschrieben und daraus dann eine Zielfunktion als Bewertungskriterium abgeleitet. Weiter ist festzulegen, welche der geometrischen Abmessungen fix vorgegeben sind und welche die zu variierenden Parameter sind. Ist alles definiert, kann mit geeigneten Optimierungsmethoden innerhalb des zulässigen Bereichs der Variablen nach dem Minimum der Zielfunktion gesucht werden.

3.1 Formoptimierung mit Dakota

Im Rahmen dieser Arbeit wird diese Formoptimierung mit der Simulationssoftware Dakota durchgeführt. Durch das Variieren der Geometrie sollen die Spannungen und Durchbiegungen eines Bauteiles optimiert werden, deshalb kommen diese beiden Größen in der Zielfunktion vor. Der Optimieralgorithmus von Dakota benötigt daher die Information, wie sich diese Eigenschaften bei Variation der Geometrie verhalten. Zur Berechnung dieses Zusammenhanges wird das Finite-Elemente-Programm Abaqus verwendet. Um die Kommunikation zwischen den Finite-Elemente-Programm Abaqus und der Optimierungssoftware Dakota zu ermöglichen, muss die Abaqus-Berechnung auf einem parametergesteuerten Finite-Elemente - Modell des Bauteiles beruhen. Das Ziel der Formoptimierung ist es, das globale Minimum der Zielfunktion innerhalb des zulässigen Bereichs zu finden, daher wurden vorrangig globale Optimierer verwendet.

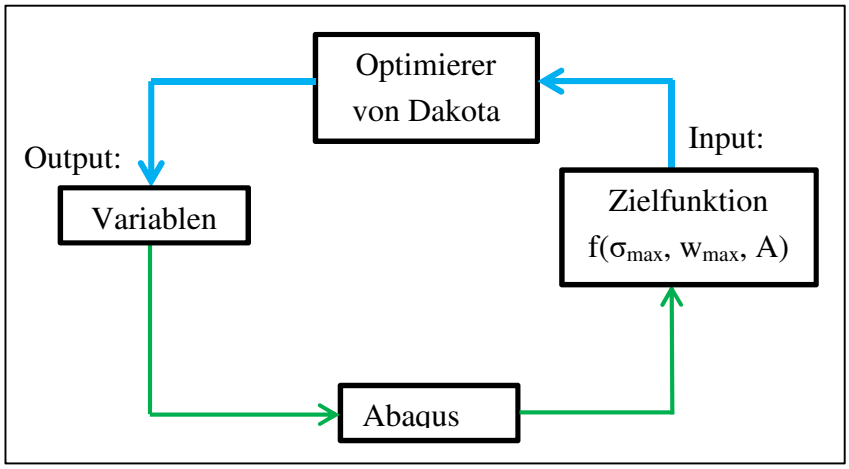


Abbildung 3.1: Ablauf der Formoptimierung

Unabhängig von der gewählten Optimierungsmethode, wird das Optimum iterativ gesucht. In Abbildung 3.1 ist der Informationsfluss innerhalb eines Iterationsschritts dargestellt. Am

Anfang jedes Iterationsschrittes gibt der Optimierer Werte für die veränderlichen Abmessungen innerhalb des zulässigen Bereiches an, für welchen er als Input den dazugehörigen Zielfunktionswert benötigt. Da unter anderem auch die Spannung und die Durchbiegung optimiert werden sollen, kommen diese auch in der Zielfunktion vor. Pro Zielfunktionsauswertung muss daher eine Abaqus-Berechnung gestartet werden, bei welcher für ein Modell, welches die vom Optimierer vorgegebenen Abmessungen hat, die maximale Spannung und die maximale Durchbiegung berechnet werden. Anschließend wird aus diesen Ergebnissen die Zielfunktion berechnet und ihr Wert an Dakota zurückgegeben. Dieser Ablauf wird so lange wiederholt, bis das Abbruchkriterium des Optimierers erfüllt ist und dadurch im Idealfall das Optimum gefunden wurde.

3.1.1 Die Dakota-Input-Datei

Der Ablauf der Optimierung wird in der Dakota-Input-Datei, welche aus sechs Blöcken besteht, festgelegt [1]. Jeder dieser Blöcke ist durch ein Schlagwort gekennzeichnet und nach diesem benannt. Sie heißen: *environment*, *method*, *model*, *variables*, *interface* und *responses*. Der Zusammenhang dieser Blöcke und damit der Aufbau der Optimierung ist in Abbildung 3.2 dargestellt. Im Anhang A1 ist die Dakota-Input-Datei von der I-Träger Optimierung zu finden. Außerdem wird mithilfe dieser Datei auch die Optimierung gestartet.

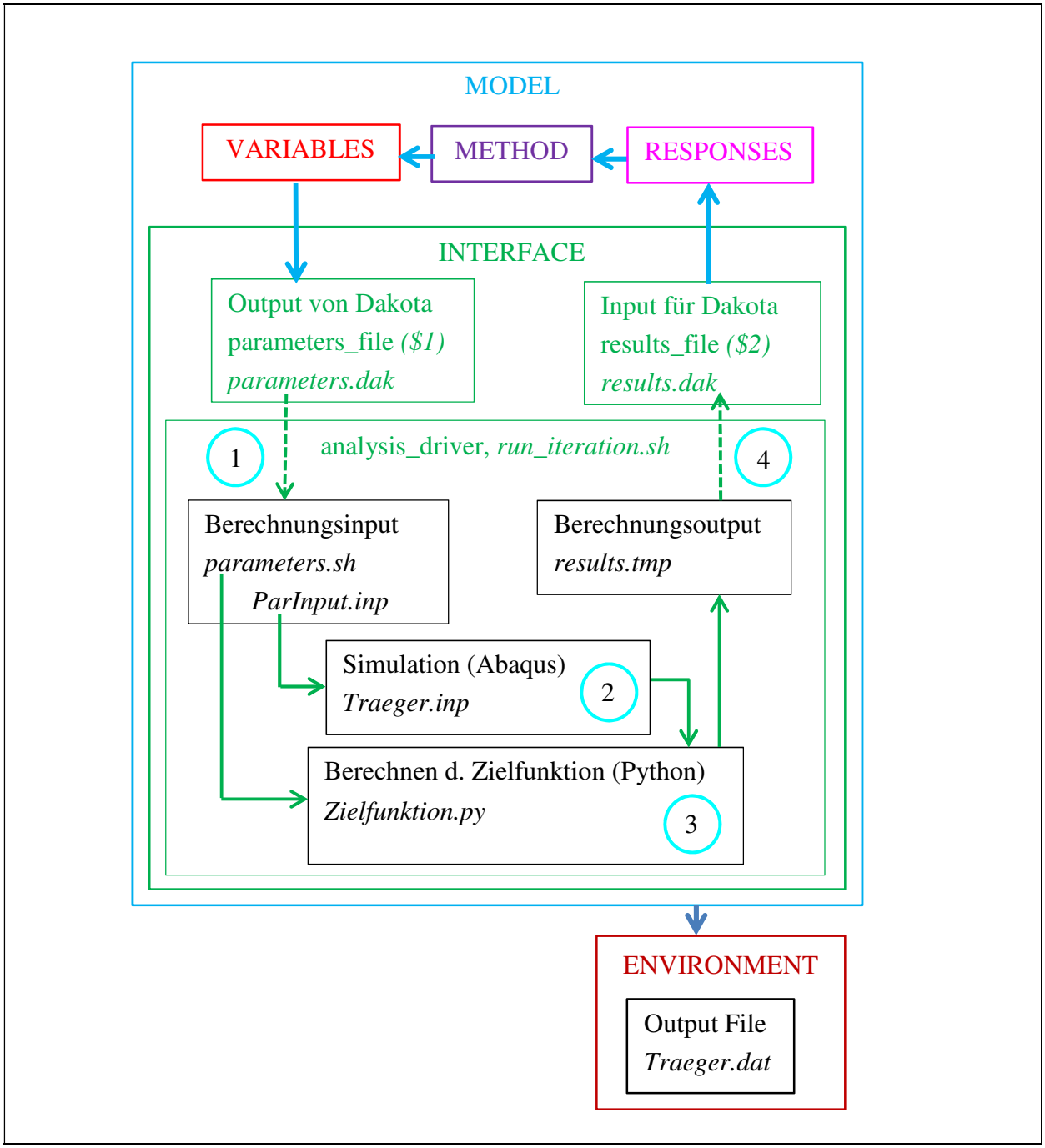


Abbildung 3.2: Aufbau der Optimierung und Zusammenhang der Blöcke *environment*, *method*, *model*, *variables*, *interface* und *responses* von der Dakota-Input-Datei

Method Block

Welche Optimierungsmethode verwendet werden soll, wird im *method* Block definiert. Auch die Eigenschaften vom Optimierer, wie zum Beispiel die Abbruchkriterien, werden hier vorgegeben.

Variables Block

Der *variables* Block ist dafür da, Anzahl, zulässigen Bereich, sowie die Eigenschaften der veränderlichen Variablen, welche für die Iteration benötigt werden, zu definieren. Im Falle

einer Formoptimierung ist hier also festgelegt, welche Abmessungen optimiert werden und deren zulässiger Bereich.

Responses Block

Im *responses* Block ist definiert, welche Daten der *method* Block am Ende jedes Iterationsschritts als Input benötigt, um die Optimierung durchführen zu können. Bei einer Formoptimierung ist das der Wert für die Zielfunktion. Falls auch noch Nebenbedingungen vorhanden sind, müssen deren Werte ebenfalls übermittelt werden. Unter *objective_funktions* wird die Anzahl der Zielfunktionen festgelegt [13]. Im Falle von Restriktionen muss hier auch angeben werden, ob es sich um Gleichungen und Ungleichungen handelt, genauso wie deren Anzahl und die dazugehörigen Limits.

Model Block

Im *model* Bock wird festgelegt, wie viele *variables*, *method, interface* und *responses* Blöcke benötigt werden und wie sie zusammengehören. Hier wird also festgelegt, welcher *interface* Block und damit welche Berechnungen, mit welchen *variable*s Block, also welchen Variablen durchgeführt werden, damit man das im *responses* Block definierte Ergebnis für den dazugehörigen *method* Block bekommt. *Single* kennzeichnet den einfachsten Fall, wo jeder dieser Blöcke nur einmal vorkommt und folglich die vier vorhandenen Blöcke zusammengehören [13].

Interface Block

Im *interface* Block wird festgelegt, wie man von den im *variables* Block definierten Parametern, zu dem im *responses* Block definierten Input für den *method* Block kommt. Im Fall von einer Formoptimierung, wird mit dem Schlagwort *fork* festgelegt, das nicht Dakota den benötigten Input berechnet, sondern, dass dies ein externes Programm übernimmt. Es handelt sich also um eine Blackbox Optimierung, denn Dakota kennt weder die analytische Form der Zielfunktion, noch die der Restriktionen. Es weiß auch nicht, welche Rechenschritte notwendig sind, um sie zu ermitteln. Unter *analysis_driver* wird festgelegt, welches Programm oder welche Datei Dakota ausführen muss, um den erforderlichen Input für den *method* Block zu ermitteln.

Die Kommunikation zwischen Dakota und dem externen Programm erfolgt über die Schlagwörter *parameters_file* und *results_file*. Das *parameters_file* ist dafür da, die Parameter vom *variables* Block an den *analysis_driver* zu übermitteln. Mit *$1* kann der Inhalt vom *parameters_file* aufgerufen werden. Im Gegenzug wird das Ergebnis vom *analysis_driver* mithilfe des *results_file* an Dakota übermittelt. Der Inhalt vom *results_file* kann mit *$2* aufgerufen werden und muss mit dem im *responses* Block definierten Input für den Optimierer zusammenpassen. *File_save* und *file_tag* sorgen dafür, dass alle während einer Iteration erzeugten Dateien unter dem im Input File definierten Namen, ergänzt mit der Nummer von der Iteration/ Funktionsauswertung, abgespeichert werden. Bei der Form-optimierung wird durch das Schlagwort *work_directoriy* für jeden Iterationsschritt ein Ordner

angelegt, welcher die unter *copy_files* angeführten Dateien enthält. In diesem Ordner wird die Berechnung der Zielfunktion durchgeführt. *Directory_tag* sorgt dafür, dass im Ordnernamen die Iterationsnummer vorkommt, und aufgrund von *directory_save* wird der Ordner nach Beendung des Iterationsschrittes nicht sofort wieder gelöscht [13].

Environment Block

Der *environment* Block ist optional, hier wird definiert, wie das Ergebnis der Optimierung abgespeichert wird. Wenn *tabular_data* ausgewählt ist, werden von jedem Iterationsschritt die Parameter, die dazugehörigen Zielfunktionswerte und die Restriktionen in einer Tabelle abgespeichert [13].

3.1.2 Ablauf in Interface Block

Der Ablauf innerhalb des *analysis_driver* lässt sich bei der Formoptimierung in vier Schritte unterteilen. Im Zuge dieser Arbeit hat er den Namen *run_iteration.sh* und die Textdatei ist im Anhang abgebildet. Der Zusammenhang der Schritte wird in Abbildung 3.2 veranschaulicht. Bei Schritt 1 wird das Preprocessing der Daten durchgeführt, dann folgt Schritt 2, die Abaqus-Berechnung, im Schritt 3 wird der Zielfunktionswert berechnet und im Schritt 4 wird das Postprocessing Daten durchgeführt.

Schritt 1: Das Preprocessing

Zu Beginn müssen, die von Dakota generierten Werte für die Parameter, welche im *parameters_file* gespeichert sind, in ein Format gebracht werden, so dass Abaqus und das Python Skript *Zielfunktion.py*, welches die Zielfunktion berechnet, damit arbeiten können. Dies geschieht mit dem von Dakota Team entwickelten Befehl *dprepro*. Für die Abaqus-Berechnung wird die Datei *ParInput.inp* generiert und für die Zielfunktionswertberechnung die Datei *parameters.sh*.

Schritt 2: Die Abaqus-Berechnung

Nach dem Preprocessing wird die Abaqus-Berechnung gestartet, dafür benötigt werden die zuvor mit *dprepro* erstellte Datei *ParInput.inp*, welche die von Dakota generierten Werte für die Variablen beinhaltet, und die Abaqus-Input-Datei des Bauteile, welche das Finite-Elemente-Modell beinhaltet. Der Aufbau dieser Dateien wird in Kapitel 3.2 beschrieben. Nach Abschluss der Berechnung erstellt Abaqus einige Dateien, für die Optimierung interessant sind die Textdatei *Abaqus_Ergebnis.dat* und die Datei *Abaqus_Ergebnis.odb*, welche die in der Abaqus-Input-Datei definierten Spannungen und die Verschiebungen beinhalten.

Schritt 3: Berechnung der Zielfunktion

Für die Berechnung des Zielfunktionswertes werden die maximale Spannung, bzw. die maximale Durchbiegung benötigt. Die Vorgehensweise, mit welcher die maximalen Werte von Abaqus für Dakota in diesem Schritt aufbereitet werden, hängt davon ab, welche von den

beiden erwähnen *Abaqus_Ergebnis* Dateien verwendet werden. In Anschnitt 3.4.1 wird näher darauf eingegangen. Mit den Parameterwerten aus der *parameter.sh* Datei und den maximalen Werten der Abaqus-Berechnung wird die Zielfunktion mit der Datei *Zielfunktion.py* berechnen. Zum Schluss sorgt das Python Skript dafür, dass der Wert der Zielfunktion und falls es Restriktionen gibt, auch der Wert für die limitierte Größe, in die Datei *results.tmp* geschrieben werden.

Schritt 4: Postprocessing

Jetzt müssen der Zielfunktionswert und der Wert für die Limitierungen nur noch dem Optimierer von Dakota übergeben werden. Das geschieht, indem der Inhalt von *results.tmp* in *results.dak* überschrieben wird.

3.2 Abaqus-Berechnung

Die Festigkeits- und Verformungsuntersuchungen werden mit dem Finite-Elemente-Programm Abaqus durchgeführt. Das Finite-Elemente-Netz ist so aufgebaut, dass es bei Änderungen der Querschnittsabmessungen entsprechend skaliert wird. Die Elementanzahl bleibt unverändert, wobei sich nur die Größe der Elemente ändert.
Die Modellgeometrie, die Netzgeometrie und die Randbedingungen sind in der Abaqus-Input-Datei festgelegt. Die Datei besteht aus sechs Abschnitten, der Parameterdefinition, der Knotendefinition, der Elementdefinition, der Materialdefinition, den Randbedingungen und Ausgaben. Im Zuge der Parameterdefinition sind jedoch nur die Werte für die konstanten Abmessungen direkt in der Input-Datei definiert. Die Variablen werden aus der Datei *ParInput.inp* eingelesen. Das vereinfacht die Kommunikation zwischen Dakota und Abaqus, welche wie folgt funktioniert. Je Iterationsschritt definiert der Optimierer von Dakota Werte für die Variablen, und die Datei *run_iteration.sh* schreibt diese in die Textdatei *ParInput.inp*. Anschließend wird die Abaqus-Berechnung gestartet und die Abaqus-Input-Datei liest die Parameter aus der Datei *ParInput.inp* ein.
Die Tatsache, dass in der Input-Datei die Lage der Knoten in Abhängigkeit von den variablen und fixen Parametern definiert werden, ermöglicht, dass das Netz mit den Querschnittsabmessungen mitskaliert wird. Danach müssen die Knoten den Elementen zugewiesen werden. Das und die Festlegung der Elementtypen geschieht im Abschnitt Elementdefinition und bleibt für alle Iterationen unverändert.
Die Finiten-Element-Modelle der Bauteile sind in dieser Arbeit zum größten Teil mit dreidimensionalen achtknotigen Hexaeder Kontinuumselementen aufgebaut. Wie alle dreidimensionalen Kontinuumselemente haben diese nur drei Freiheitsgrade je Knoten. Das Ergebnis wird bei diesem Element in globalen Koordinatenrichtungen ausgegeben, da kein lokales Koordinatensystem bei den Elementen hinterlegt ist. Weil diese Elemente nur translatorische Freiheitsgrade besitzen, kann an ihnen jedoch kein Moment angreifen. Deshalb kommen, je nach Beispiel, noch Balkenelemente bzw. Schalenelemente zum Einsatz um Momente einleiten zu können.

Balkenelemente sind eindimensionale Strukturelemente. Sie besitzen drei translatorische und drei rotatorische Freiheitsgrade und werden verwendet, wenn der Querschnitt eines Trägers im Vergleich zu seiner Länge klein ist. Der Knotenabstand bestimmt die Länge des Elements. Die weiteren Abmessungen werden über Eingabe der Querschnittsdaten festgelegt. Bei den verwendeten Elementen handelt es sich um Balkenelemente nach der Euler-Bernoulli-Theorie. Dabei wird angenommen, dass die Balkenquerschnitte, die vor der Deformation senkrecht zur Stabachse stehen auch nachher senkrecht zur deformierten Stabachse sind und dass die Querschnitte ebene bleiben.

Schalenelemente sind Strukturelemente welche verwendet, wenn die Dicke eines Körpers deutlich kleiner ist als die anderen Abmessungen. Die Elemente werden in einer Referenzfläche definiert, welche zwischen den Knoten aufgespannt wird. Die Dicke der Schale wird dann noch zusätzlich angegeben werden. Sie beruhen auf der Kirchhoff´schen Schalentheorie. Wie auch die Balkenelemente, haben die verwendeten Schalenelemente sechs Freiheitsgrade.

Unabhängig von der Art der Elemente, werden der Elementtyp, die Anzahl der Knoten, die Art und Anzahl der Freiheitsgrad und die Anzahl der Integrationspunkte durch den Namen der Elemente eindeutig festgelegt [14]. In Abbildung 3.3 ist die Bezeichnung für das Kontinuums-element C3D8R beispielhaft dargestellt.

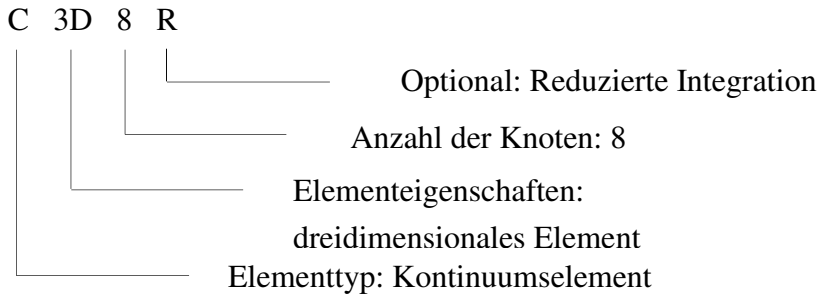


Abbildung 3.3: Beispiel für die Namen Konvention von Abaqus-Elementen[14]

Abaqus Elemente, welche für Festigkeits- und Verformungsuntersuchungen verwendet werden, beruhen auf der Lagrangeschen Betrachtungsweise. Wie schon im Kapitel „Theoretischer Hintergrund“ erwähnt, werden bei einer Analyse die Verschiebungen der Knoten, entsprechend der Freiheitsgrade, welche dieser besitzt, berechnet. Das restliche Verschiebungsfeld eines Elements wird interpoliert. Für die numerische Berechnung des Materialverhaltens, also die Berechnung von den Spannungen und Verzerrungen an den Integrationspunkten, wird bei Abaqus bei den meisten Elementen die Gauß´sche Integration verwendet. Bei einigen Kontinuumselementen gibt es die Möglichkeit zwischen exakter und reduzierter Integration zu wählen. Bei Elementen mit reduzierter Integration wird bei der Berechnung der Elementsteifigkeit eine geringere Integrationsordnung verwendet. Die Massenmatrix und die verteilten Belastungen werden jedoch voll Integriert. Durch die reduzierte Integration wird weniger Rechenzeit benötigt und die Ergebnisse verbessert.

3.3 Ablage der Daten

Vor Beginn der Berechnung befinden sich nur der Ordner *DakotaInput* mit der Datei *Traeger.in* und der Ordner *Job* mit den Dateien *run_iteration.sh, Traeger.inp, Zielfunktion.py, dprepro.perl, parameters_template.sh und ParInput_template.inp* im Arbeitsverzeichnis. Diese Dateien sind in Abbildung 3.4 grün umrahmt.

Traeger.in ist die Dakota-Input-Datei, welche benötigt wird, um die Optimierung zu starten und in welcher alle Optimierungsparameter gespeichert sind. *Run_iteration.sh* ist der *analysis_driver.* Das ist jene Datei, welche die Abaqus-Berechnung über die Datei *Traeger.inp* und die Zielfunktionsberechnung mit der Datei *Zielfunktion.py* startet. Weiters führt diese Datei auch das Pre- und Postprocessing für die beiden vorhin genannten Berechnungen aus und zwar mit *dprepro.perl* und mit den beiden dazugehörigen Template-Dateien *parameters_template.sh* und *ParInput_template.inp*.

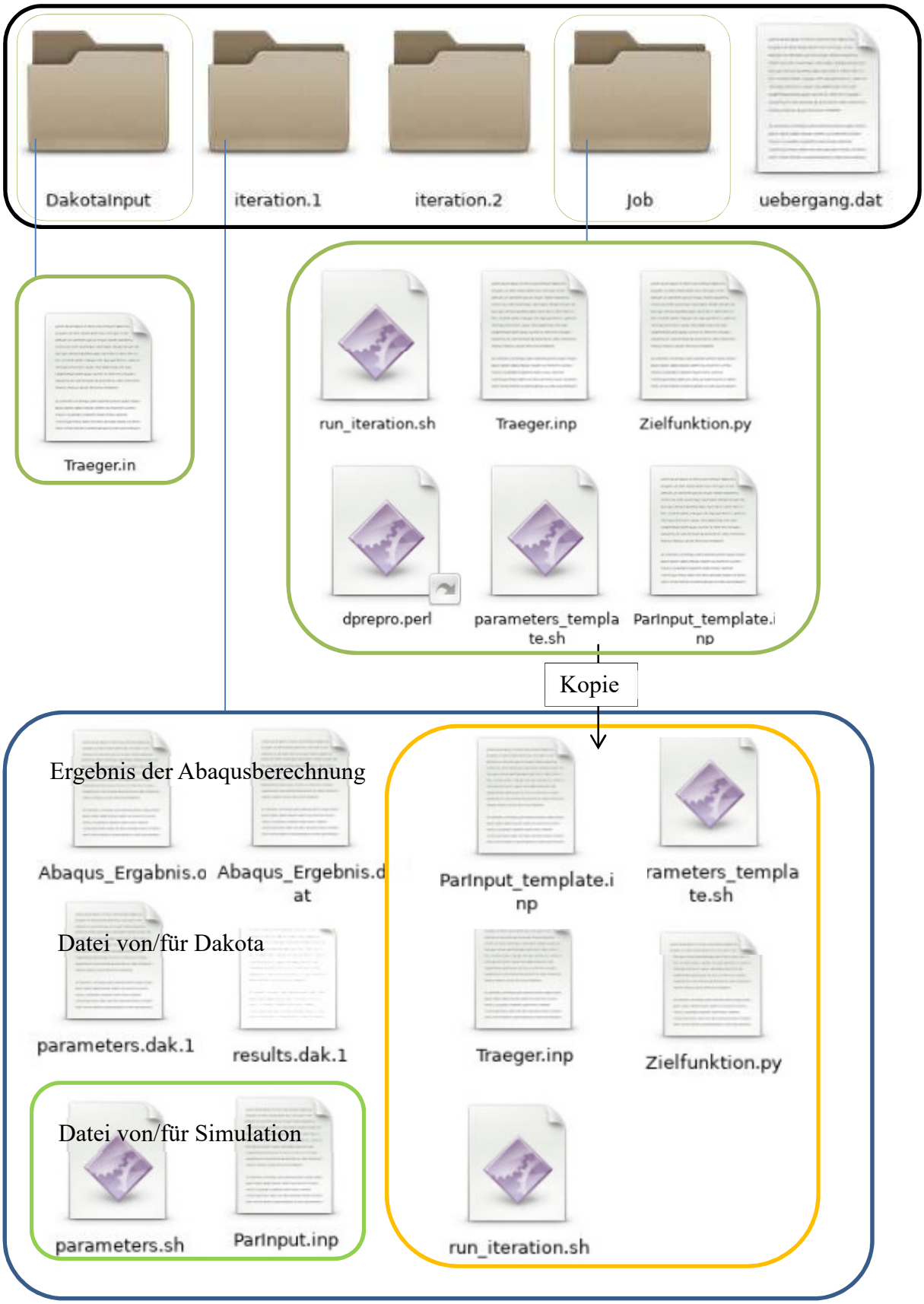


Abbildung 3.4: Ablage der Dateien

Nach dem Start der Optimierung mit *dakota –i DakotaInput/Traeger.in* wird zu Beginn jedes Iterationsvorganges ein Ordner angelegt, dessen Name sich aus dem Wort Iteration und einer fortlaufenden Nummer zusammensetzt. Das geschieht, weil in der Input-Datei die beiden Befehle *directory_save* und *directory_tage* aktiv sind. Wenn es sich um die erste Iteration handelt, heißt der Ordner *Iteration.1*. Zuerst werden alle Dateien, bis auf die Datei *dprepro.perl* aus dem Ordner *Job,* in diesen Ordner hineinkopiert. Hinzu kommt noch die von Dakota generierten Output Datei *parameters.dak.1*. Die Datei *run_iteration.sh* sorgt anschließend dafür, dass in diesem Ordner die Berechnungen für den jeweiligen Iterationsschritt durchgeführt werden. Im Rahmen dieser Berechnung werden zuerst aus der Datei *parameters.dak.1* mit *dprepro* die Input-Dateien *parameters.sh* und *ParInput.inp* erstellt. Nach Abschluss der Finite-Elemente-Berechnung generiert Abaqus einige Dateien, die für die Optimierung interessant sind. Das sind die Dateien *Abaqus_Ergebnis.odb* und *Abaqus_Ergebnis.dat*, welche das Berechnungsergebnis enthalten. Die Datei *run_iteration.sh* startet nicht nur die Abaqus-Berechnung, sondern sorgt auch dafür, dass jene maximalen Werte von der Spannung und der Durchbiegung, welche für die Zielfunktionswertberechnung benötigt werden, in die Datei *result.tmp* geschrieben werden. Anschließend wird mit dem Python Skript *Zielfunktion.py* und den Dateien *result.tmp* und *Parameters.sh* die Zielfunktion berechnet und falls Limitationen vorhanden sind, die entsprechenden Werte dafür. Der Inhalt von *result.tmp* wird jetzt mit den Werten der Zielfunktion und den Werten der zu limitierenden Größen überschrieben. Damit Dakota diesen Inhalt einlesen kann, wird dieser zum Schluss in die Datei *results.dak.1* übertragen. Jetzt ist ein Iterationsschritt abgeschlossen und falls das Abbruchkriterium vom Optimierer noch nicht erfüllt wird, wird ein neuer Ordner für eine neue Iteration erstellt. Zusätzlich wird im Arbeitsordner neben den Iterations Ordnern auch noch die Datei *Traeger.dat* erstellt, in welcher in Tabellenform für jeden Iterationsschritt die Werte der Parameter und die dazugehörigen Zielfunktionswerte abgespeichert werden.

3.4 Verbesserung der Optimierung

Es wurde sich als Ziel gesetzt, dass bei dem in dieser Arbeit behandelten I-Träger Optimierungsbeispiel mit nur einer Optimierungsvariable der relative Fehler für die Optimierungsvariable z kleiner als 1% sein soll, bezogen auf die analytische Lösung. Da dies bei den ersten durchgeführten Optimierungen nicht der Fall war, wurden zwei mögliche Verbesserungspotenziale untersucht. Zuerst wurde überprüft, ob das Optimierungsergebnis verbessert werden kann, wenn mehr Nachkommastellen von den Spannungs- und Verschiebungswerten an Dakota übergeben werden. Ferner hat man auch den Einfluss von unterschiedlichen Optimierungsalgorithmen untersucht.

3.4.1 Einlesen der Abaqus-Ergebnisse verbessern

Von der Art der Einlesevariante der Abaqus Ergebnisse hängt ab, auf wie viele Nachkommastellen genau das Ergebnis von der Abaqus-Berechnung an Dakota übermittelt wird. Um die Einlesegenauigkeit zu ändern müssen zwei Textdateien modifiziert werden, der

analysis_driver run_iteration.sh und die Datei *Zielfunktion.py*. Drei unterschiedliche Varianten werden verglichen. Pro Variante gibt es eine Version von den beiden Dateien und diese sind im Anhang A.2 dargestellt. Bei der ursprünglich verwendeten Variante „Text 1“ wird die Spannung auf eine Nachkommastelle und die Durchbiegung auf drei Nachkommastellen genau angegeben. Bei der Variante „Text 2“ erhöht sich bei der Durchbiegung die Anzahl der Nachkommastellen von drei auf sieben. Bei der Variante „odb“ wird die Spannung auf sieben und die Durchbiegung auf neun Nachkommastellen genau angegeben. Der in diesem Kapitel angeführte Vergleich hat gezeigt, dass die „odb“ Variante zu den besten Optimierungsergebnissen führt. Daher wird sie für alle weiteren Berechnungen verwendet.

Einlesevariante Text 1

Das Einlesen der maximalen Werte der Finite-Elemente-Berechnung erfolgt über die von Abaqus erstellte Text Datei *Abaqus_Ergebnis.dat*. Ein Ausschnitt dieser Datei ist im Anhang A.2 Abbildung A. 2 dargestellt. Der maximale Spannungswert und die maximale Durchbiegung sind in dieser Datei über das Schlagwort *maximum* zu finden. Diese Tatsache wird ausgenutzt und in der Textdatei *run_iteration.sh,* welche im Anhang A2 Abbildung A. 3 dargestellt ist, mit den Befehlen *grep* und *sed* nach diesem Schlagwort gesucht und die maximalen Werte in *results.tmp* geschrieben. Anschließend wird mit der Datei *Zielfunktion.py*, welche im Anhang A.2 Abbildung A. 4 zu finden ist, und den Werten aus *results.tmp* der Zielfunktionswert berechnet.

Einlesevariante Text 2

Wenn man die Datei *Abaqus_Ergebnis.dat* im Anhang A.2 Abbildung A. 2 genauer betrachtet, erkennt man, dass im allgemeinen Teil der Tabelle, in der die Knotenpunktverschiebungen angeführt sind, die Verschiebungen auf sieben Nachkommastellen genau angegeben werden. Bei der Zusammenfassung am Ende der Tabelle, wo der maximale Wert angeführt ist, wird dieser jedoch nur auf drei Nachkommastellen genau angegeben. Die Spannungswerte sind sowohl in der Tabelle, als auch in der Zusammenfassung nur auf eine Nachkommastelle genau angegeben.
Bei den in dieser Arbeit untersuchten Bauteilen tritt unabhängig von den veränderten Querschnittsabmessungen die maximale Durchbiegung, immer im gleichen Knoten auf. Um die Genauigkeit der Optimierung zu erhöhen, wurde bei der Variante „Text 2“ zusätzlich zu dem in der Zusammenfassung angeführten maximalen Wert der Durchbiegung auch noch der, in der Tabelle angeführte auf sieben Nachkommastellen genaue Wert von der maximalen Durchbiegung eingelesen.
Um sicher zu gehen, dass auch wirklich immer die maximale Durchbiegung verwendet wird, werden in der Datei *Zielfunktion.py* die beiden eingelesenen Werte der maximalen Durchbiegung verglichen und nur, wenn die Differenz der beiden kleiner als 0,001 mm ist, wird der auf sieben Nachkommastellen genaue Wert verwendet. Die *run_iteration.sh* Datei

von der Variante „Text 2“ ist im Anhang A.2 Abbildung A. 5 zu finden und die dazugehörige Datei *Zielfunktion.py* im Anhang A.2 Abbildung A. 6.

Einlesevariante odb

Da in der Datei *Abaqus_Ergebnis.dat* die Spannungen immer nur auf eine Nachkommastelle genau angegeben werden, wird als Alternative das Einlesen der maximalen Werte aus der von Abaqus nach der Finiten-Elemente-Berechnung erstellten odb-Datei *Abaqus_Ergebnis.odb* durchgeführt. Für diese Einlesevariante muss wieder die Textdatei *Zielfunktion.py* und auch die Textdatei *run_iteration.sh* geändert werden, siehe Anhang A.2 Abbildung A. 7 und Abbildung A. 8. Die Werte für die maximalen Spannungen werden jetzt auf sieben Nachkommastellen und für die maximale Durchbiegung auf neun Nachkommastellen genau angegeben.

Vergleich der unterschiedlichen Einlesevarianten

Um erkennen zu können, ob die Änderung der Einlesevariante die Optimierung verbessert hat, werden die Ergebnisse von der Formoptimierung des I-Trägers untersucht. Details zu dieser Optimierung sind in Kapitel 4.1 zusammengefasst. Es werden mit zwei unterschiedliche Zielfunktionen je drei Optimierungen für die Gurtdicke z durchgeführt. Bei den drei zusammengehörigen Optimierungen sind alle Optimierungsparameter gleich. Sie unterscheiden sich nur in der Art, wie das Ergebnis von Abaqus in Dakota importiert wird. Verglichen werden das Optimierungsergebnis z_{opti}, der dazugehörige Wert der Zielfunktion und entsprechend der gewählten Zielfunktion die ebenfalls dazugehörige maximale Durchbiegung bzw. Spannung des Bauteiles. Um eine geeignete Referenz für die Berechnung des relativen Fehlers zu haben, wurde eine Abaqus-Berechnung durchgeführt, bei welcher für z, das Ergebnis der analytischen Optimierung eingesetzt wurde. Für die Berechnungen werden folgende Abmessungen und Materialkenngrößen verwendet: $b = 20$ mm, $h = 30$ mm, $M = 625000$ Nmm, $r = 0{,}2$ mm, $l = 100$ mm, $E = 70000$ N/mm^2 und $y = 1$ mm. Bei den vorkommenden Spannungen handelt es sich immer um die maximalen Spannungen $\sigma_{x\,max}$ in den Integrationspunkten und bei den Verschiebungen um die maximale Verschiebung $w_{z\,max}$ der Knotenpunkte. Für die Finite-Elemente-Berechnung wurde das in Kapitel 4.1.3 angeführte „normale“ Netz und die ebenfalls in diesem Kapitel beschrieben KC3 Randbedingungen verwendet. Die Berechnung der analytischen Lösung ist in Kapitel 4.1.2 beschrieben. Für die Optimierung wurde der Optimierer *coliny_cobyla* verwendet.

Tabelle 3.1: Vergleich der unterschiedlichen Einlesevarianten für die Zielfunktion $w_{z\,max}\cdot A$

Zielfunktion $w_{3\,max}\cdot A$							
Einlese-variante	Ergebnis				reletiver Fehler in %		
	z_{opti} in mm	$w_{z\,max}$		$w_{z\,max}\cdot A$ in mm3	z_{opti}	$w_{z\,max}$	$w_{z\,max}\cdot A$
		in mm	Anzahl der NKST				
z = 2,26 *	2,260 *	2,392 *	-	277,15 *	-	-	-
Text 1	2,312	2,351	3	277,12	2,32	1,68	0,01
Text 2	2,259	2,392	7	277,15	0,03	0,02	0,00
odb	2,259	2,392	9	277,15	0,05	0,00	0,00

* Referenzwerte, keine Optimierungsergebnisse sondern das Ergebnis von der Abaqusberechung für z = 2,26 mm

Zuerst werden die Optimierungsergebnisse für die Zielfunktion $w_{z\,max}\cdot A$ verglichen und die Ergebnisse in der Tabelle 3.1. zusammengefasst. Beim Optimierungsergebnis von z_{opti} als auch beim dazugehörigen Wert von $w_{z\,max}$ sind signifikante Verbesserungen zu erkennen, wenn man auf sieben anstatt auf drei Nachkommastellen von $w_{3\,max}$ zugreift. Auch das Verwenden der odb-Datei führt zu einer ähnlichen Verbesserung gegenüber der Auswertung mit drei Nachkommastellen. Die Ergebnisse von den Varianten „Text 2“ und „odb“ sind gleichwertig. Die Querschnittsfläche des I-Trägers wird immer analytisch berechnet mit dem Python Skript *Zielfunktion.py*. In dieser ist die Formel für den Flächeninhalt hinterlegt. Der Wert der Fläche ist zahlenmäßig viel größer als jener der Durchbiegung und unabhängig von der Einlesegenauigkeit des Abaqus-Ergebnisses. Daher weicht der Wert der Zielfunktion bei der Variante „Text 1“ schon nur sehr wenig von der analytischen Lösung ab. Aus demselben Grund gibt es auch keine große Verbesserung, wenn andere Einlesevarianten verwendet werden.

Tabelle 3.2: Vergleich der unterschiedlichen Einlesevarianten für die Zielfunktion $\sigma_{x\,max}\cdot A$

Zielfunktion $\sigma_{x\,max}\cdot A$							
Einlese-variante	Ergebnis				reletiver Fehler in %		
	z_{opti} in mm	$\sigma_{x\,max}$		$\sigma_{x\,max}\cdot A$ in N	z_{opti}	$\sigma_{x\,max}$	$\sigma_{x\,max}\cdot A$
		in N/mm^2	Anzahl der NKST				
z = 2,26 *	2,260 *	495,2 *	-	57387,7 *	-	-	-
Text 1	2,297	489,3	1	57385,7	1,63	1,20	0,00
Text 2	2,297	489,3	1	57385,7	1,63	1,20	0,00
odb	2,259	495,5	7	57387,7	0,06	0,05	0,00

* Referenzwerte, keine Optimierungsergebnisse sondern das Ergebnis von der Abaqusberechung für z = 2,26 mm

Beim zweiten Vergleich werden die Zielfunktion $\sigma_{x\,max}\cdot A$ und die maximale Spannung $\sigma_{x\,max}$ verglichen, siehe Tabelle 3.2. Da bei den Varianten „Text 1“ und „Text 2“ die Spannung die gleiche Anzahl an Nachkommerstellen hat, sind die Ergebnisse ident. Wenn die „odb“ Einlesevariante verwendet wird und sich dadurch die Anzahl der Nachkommastellen von eins

auf sieben erhöht, ist der relative Fehler für z_{opti} und $\sigma_{x\,max}$ deutlich geringer. Aus den gleichen Gründen wie bei der Zielfunktion $w_{z\,max}$·A, ändert sich der relative Fehler der Zielfunktion nicht.
Da die Einlesevariante „odb“ die besten Ergebnisse liefert, wird sie für weitere Berechnungen verwendet.

3.4.2 Vergleich von Optimieralgorithmen

Es ist nicht jeder Optimieralgorithmus für jede Optimierungsaufgabe gleich gut geeignet. Daher werden sechs unterschiedliche Optimieralgorithmus von Dakota ausprobiert: *coliny_ea*, *soga, ncsu_direct*, *coliny_direct, efficient_global* und *coliny_cobyla. Coliny_cobyla* ist der einzige lokale Optimierer, welcher ausprobiert wird. Es werden vorrangig globale Optimierer verwendet, da nach einem globalen Optimum gesucht wird. Bei den drei Optimieralgorithmen *coliny_direct, coliny_cobyla* und *ncsu_direct* handelt es sich um deterministische Optimierungsalgorithmen. Wenn bei einer Optimierung mit deterministischem Algorithmus die Optimierungsparameter nicht geändert werden, also der Optimieralgorithmus, die dazugehörigen Einstellungen und die Startwerten gleich bleiben, wird auch bei mehrmaliger Anwendung immer die gleiche Folge an Zwischenschritten durchlaufen. Man bekommt daher immer die gleiche Endlösung. Nur durch variieren der Startwerte besteht die Möglichkeit, bei einem weiteren Durchlauf der Optimierung eine bessere Lösung zu erhalten. Die restlichen drei verwendeten Algorithmen beruhen auf stochastische Verfahren. Sie beinhalten eine stochastische Komponente, das heißt, auch bei gleichen Optimierungsparametern können die Ergebnisse geringfügig voneinander abweichen. In Kapitel 2.1.2 ist genau angeführt auf welchem Algorithmus die jeweiligen Optimierer beruhen.
Da die behandelten Beispiele eine einfache Geometrie aufweisen, sind die bei den Beispielen verwendeten Zielfunktionen konvex, und deshalb findet auch der lokale Optimieralgorithmus das Optimum, und zwar um einiges schneller als die meisten anderen. Ein weiterer sehr schnell und robust funktionierender Optimierer ist *efficient_global*. Daher sind *efficient_global* und *coliny_cobyla* die meist verwendeten Algorithmen in dieser Arbeit. In Tabelle 4.7 und Tabelle 4.6 wurden jeweils die gleichen Optimierungsaufgaben mit unterschiedlichen Optimieralgorithmen gelöst und es ist klar ersichtlich, dass *efficient_global* und *coliny_cobyla* mit Abstand am schnellsten das Optimum finden.

3.5 Adaption der Optimierung für andere Beispiele

In diesem Kapitel wird beschrieben, welche Dateien geändert werden müssen, um ein anderes Bauteil optimieren zu können. Bevor man die Dateien überarbeitet, muss man sich überlegen, welche Zielfunktion für die Optimierung geeignet ist, und welche Variablen man benötigt, sowie deren zulässigen Bereich. Daneben benötigt man eine Abaqus-Input-Datei, welche alle für die Finite-Elemente-Berechnung benötigten Daten enthält und deren Ergebnis die Spannungs- und Verschiebungswerte beinhaltet, welche für die Zielfunktion benötigt wird. Um ein parametergesteuertes Finite-Elemente-Modell zu haben, werden die veränderlichen

Variablen nicht direkt in der Input-Datei gespeichert, sondern in der externen Datei *ParInput.inp*. Damit Dakota und Abaqus kommunizieren können, müssen die Variablen und deren Platzhalter im *ParInput_template.inp* angepasst werden. Das gleiche gilt auch für die *parameters_template.sh* Datei, welche für die Datenübertragung zwischen Dakota und den Python Skript verantwortlich ist. Auf jeden Fall aktualisiert werden müssen in der Datei *run_iteration.sh* die Namen und die Anzahl der Variablen, welche nach *Zielfunktion.py* aufgelistet sind. Wenn die Namen der Abaqus-Input-Datei *Traeger.inp* oder der beiden erwähnten Template-Dateien geändert wurden, müssen diese auch in der Datei *run_iteration.sh* aktualisiert werden. Im Skript *Zielfunktion.py* wird die Zielfunktion berechnet. Hier muss das Einlesen der odb-Datei überarbeitet werden, so dass die für die Zielfunktion benötigten Verschiebungs- und Spannungskomponenten bzw. Vergleichsspannungen importiert werden können. Weiters müssen auch die Formeln für die Zielfunktionswertberechnung angepasst werden. Die für die Berechnung erforderlichen Variablen müssen mit jenen, welche in der *run_iterations.sh* Datei angeführt sind, mit jenen in der Dakota-Input-Datei und mit jenen von der *parameters_template.sh* Datei übereinstimmen.

Die letzte Datei, welche noch überarbeitet werden muss, ist die Dakota-Input-Datei *Traeger.in*. Im Block *method* müssen ein geeigneter Optimieralgorithmus und passende Abbruchkriterien ausgewählt werden. Unter *variables* müssen die Anzahl und der zulässige Bereich der Variablen angegeben werden. Ebenso überarbeitet werden muss der *responses* Block, denn hier werden, falls vorhanden, die Limits für die Spannungen und die durch Biegungen definiert. Falls die Namen der Dateien geändert wurden, müssen die Namen im *interface* Block auch aktualisiert werden.

4 Numerische Testfälle

Anhand von zwei Beispielen wird in dieser Arbeit untersucht, ob und wie die Formoptimierung mit Dakota und Abaqus umgesetzt werden kann. Für einen I-Träger wird das optimale Profil sowohl analytisch als auch numerisch mit Dakota und Abaqus ermittelt. Durch diese Vorgehensweise können die Ergebnisse verglichen und überprüft werden. In einem nachfolgenden Beispiel wird die Form eines Sandwichkeils untersucht.

4.1 I-Träger

Aufgrund seines I-förmigen Querschnittsbesitzt der I-Träger ein großes Flächenträgheitsmoment bei geringem Gewicht. Die horizontalen Anteile des Querschnittes heißen Gurte und das vertikale Abschnitt Steg.
Für den in Abbildung 4.1 dargestellten I-Träger sollen Querschnittsabmessungen gefunden werden, für welche das Gewicht, die Durchbiegung und die maximal auftretenden Spannungen minimal sind. Der Träger ist auf einer Seite fest eingespannt und auf der anderen wird er mit einem Biegemoment belastet. Diese Formoptimierung soll sowohl analytisch als auch numerisch mit Abaqus und Dakota durchgeführt werden. Der Zweck der analytischen Lösung ist es, die numerischen Ergebnisse zu verifizieren.

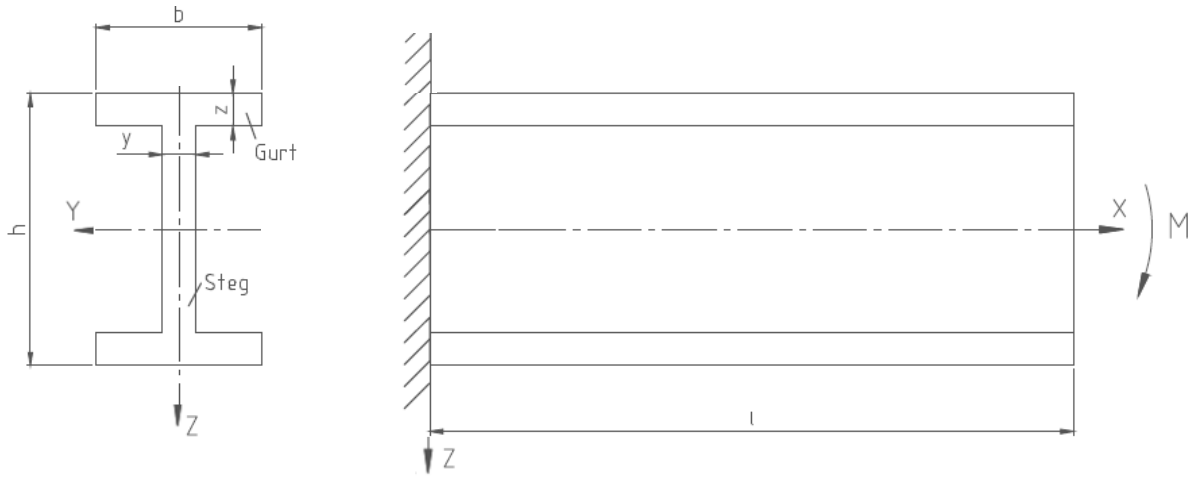


Abbildung 4.1: Geometrie und Bemaßung des Testbeispiels I-Träger

Der I-Träger besteht aus isotropem Material dessen Eigenschaften durch den E-Modul E, die Poissonzahl ν und die Dichte ρ festgelegt sind. Die Geometrie des Trägers wird definiert mit der Gurtdicke z, der Steghöhe h, der Trägerbreite b, der Trägerhöhe h und der Trägerlänge l. Ferner werden für die Formoptimierung der Randfaserabstand e, das Flächenträgheitsmoment I_y um die y-Achse und die Querschnittsfläche A benötigt. Belastet wird der Träger, wie in Abbildung 4.1 dargestellt, mit dem Biegemoment M. Eine weitere benötigte Größe ist die Schichtdicke r der Gurtelemente an der Außenseite, die in Kapitel 4.1.3 beschrieben wird.

Folgende Werte werden für die nachfolgenden Berechnungen verwendet: $b = 20\,mm$, $h = 30\,mm$, $M = 625000\,Nmm$, $r = 0{,}2\,mm$, $l = 100\,mm$, $E = 70000\,N/mm^2$, $\nu = 0{,}33$, $\rho = 2700\,kg/m^3$, $y = 1\,mm$ und $z = 2\,mm$.

4.1.1 Festlegen der Optimierungsvariablen und des Optimierungsziels

Die Querschnittsabmessungen sind definiert über die Gurtdicke z, die Stegdicke y, die Trägerhöhe h und die Trägerbreite b. Diese vier Größen kommen daher als Optimierungsvariablen für die Formoptimierung in Frage. Um herauszufinden, welche dieser Variablen sich am besten für die Optimierung eignen und um ein sinnvolles Optimierungsziel zu definieren, werden die Formeln für die zu minimierenden Größen, das sind die Masse, die Spannung und die Durchbiegung, genauer betrachtet.

Aus der Formel für die Masse m, ist ersichtlich, dass sie nur von der Querschnittsfläche A abhängt, da l und ρ vorgegeben konstante Größen sind. Das Gewicht wird minimal, wenn die Querschnittsfläche minimal ist.

$$m = A \cdot l \cdot \rho \tag{21}$$

Da es zu aufwendig ist, für unterschiedliche Querschnittsabmessungen jeweils alle Spannungs- bzw. Verformungszustände des Bauteiles zu vergleichen, werden stellvertretend dafür nur deren maximale Werte betrachtet. Die maximale Durchbiegung tritt, wie in Abbildung 4.2 dargestellt, in positive z-Richtung und am oberen Ende des Trägers auf. Für die Berechnung der maximalen Durchbiegung w_{max} des gesamten Trägers werden, wie in (24) ersichtlich, die Durchbiegung der Stabachse w_z und die dazugehörige Winkeländerung φ der Stabachse am Ende des Trägers benötigt.

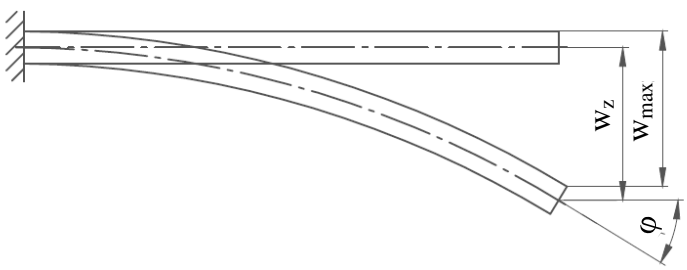


Abbildung 4.2: Darstellung der maximalen Durchbiegung der I-Trägers

$$w_z = \frac{M \cdot l^2}{2 \cdot E \cdot I_y} \tag{22}$$

$$\varphi = \frac{M \cdot l}{E \cdot I_y} \tag{23}$$

$$w_{max} = w_z + \frac{h}{2} - \cos(\varphi) \cdot \frac{h}{2} \tag{24}$$

Da die Winkeländerung φ sehr klein ist, gilt:

$$\frac{h}{2} - \cos(\varphi) \cdot \frac{h}{2} \ll w_z \tag{25}$$

Aufgrund des geringen Unterschiedes zwischen w_z und w_{max} und um die Formel zu vereinfachen, wird bei der analytischen Optimierung w_z anstelle von w_{max} verwendet. Durch diese Vereinfachung sind die möglichen Optimierungsvariablen in der Formel für die Durchbiegung nur noch im Flächenträgheitsmoment I_y um die y-Achse enthalten. Die Durchbiegung wird minimal, wenn dieses Flächenträgheitsmoment maximal wird.

Die dominierenden Spannungen, sind die Normalspannungen σ_x in x-Richtung. Die maximale Spannung $\sigma_{x\,max}$ tritt in der Randfaser bei dem Randfaserabstand e = h/2 auf.

$$\sigma_{x\,max} = \frac{M}{I_y} \cdot e = \frac{M}{I_y} \cdot \frac{h}{2} \tag{26}$$

Die Querschnittsabmessungen fließen über das Flächenträgheitsmoment I_y und über die Höhe h in die Spannung ein.

Damit die möglichen Optimierungsvariablen in den zu optimierenden Größen nur noch in der Querschnittsfläche und dem Flächenträgheitsmoment vorkommen, wird die Höhe h nicht als Optimierungsvariable gewählt. Es können jedoch nicht zeitgleich alle drei Größen, die Spannung, die Durchbiegung und die Masse, ihren kleinsten Wert annehmen. Denn die Masse wird minimal, wenn der Querschnitt klein ist, und die Spannung und die Durchbiegung sind minimal, wenn das Flächenträgheitsmoment groß ist.

Ein mögliches Optimierungsziel um die Masse m, die maximale Durchbiegung des Trägers $w_{z\,max} = w_z$ und die maximale Spannung des Trägers $\sigma_{x\,max}$ zu minimieren ist:

$$\begin{aligned} w_{z\,max} \cdot m &\rightarrow min \\ \sigma_{x\,max} \cdot m &\rightarrow min \end{aligned} \tag{27}$$

Wenn man bei der Durchbiegung den Term mit Winkeländerung φ vernachlässigt und die Trägerhöhe h als konstant annimmt, erkennt man, an den die Formeln (24) und (26), dass sich $w_{z\,max} \cdot m$ und $\sigma_{x\,max} \cdot m$ nur noch in den Konstanten l, E und h unterscheiden. Da konstante Größen die Optimierung nicht beeinflussen, erhält man für beide Zielfunktionen das gleiche Optimierungsergebnis. Die Zielfunktion kann vereinfacht werden zu:

$$\frac{m}{I_y} \rightarrow min \tag{28}$$

Da die Länge l und die Dichte ρ konstant sind, kann die Masse m durch die Querschnittsfläche A ersetzt werden. Damit wird

$$\frac{A}{I_y} \rightarrow min \tag{29}$$

zum geeigneten Optimierungsziel.

Für die analytische Berechnung wird A / I_y als Zielfunktion gewählt, wodurch sich der Schreibaufwand reduziert. Für die numerische Optimierung wird $w_{z\,max} \cdot$ A bzw. $\sigma_{x\,max} \cdot A$ als Zielfunktion verwendet, da Abaqus die Werte für $\sigma_{x\,max}$ und $w_{z\,max}$ ausgibt.

Als Optimierungsvariablen stehen z, y und b zur Verfügung. Da die Optimierung sowohl analytisch als auch numerisch durchgeführt werden soll, können die genauen Optimierungsvariablen erst festgelegt werden, wenn überprüft wurde, ob für die gewählte Konstellation eine analytische Lösung vorhanden ist.

4.1.2 Analytische Optimierung

Um die numerische Lösung der Formoptimierung des I-Trägers überprüfen zu können, wird die analytische Lösung dieser Optimierung benötigt. Das Optimierungsziel dieser ist, ein Minimum für die Funktion Querschnittsfläche A durch Flächenträgheitsmoment I_y um die y-Achse des I-Trägers zu finden. Die Zielfunktion lautet wie folgt:

$$F = \frac{A}{I_y} = \frac{2bz + y(h-2z)}{2\left(\frac{bz^3}{12} + bz\left(\frac{h-z}{2}\right)^2\right) + \frac{y(h-2z)^3}{12}} \tag{30}$$

Um sicherzustellen, dass die Lösung geometrisch möglich ist, sind folgende zulässigen Bereiche für die Optimierungsvariablen einzuhalten:

$$\begin{gathered} 0 < z < \frac{h}{2} \\ 0 < b \\ 0 < y < b \end{gathered} \tag{31}$$

Unabhängig davon, welche der drei möglichen Optimierungsvariablen man für eine Optimierung mit zwei Variablen verwendet, gibt es keine analytische Lösung. Daher wird das Problem auf eine Variable reduziert, indem man z und y gleichgesetzt hat. Für dieses Optimierungsproblem gibt es jedoch auch keine analytische Lösung. Es wird erst eine Lösung gefunden, wenn y als konstant angenommen wird und z die einzige Variable ist, also für die Zielfunktion $F(z)$.

Analytische Lösung für $F(z)$

Mit Hilfe der Ableitungen der Zielfunktion $F(z)$ nach z wurden ein lokales Minimum z_3 und zwei lokale Maxima z_1 und z_2 ermittelt. Die beiden Maxima liegen außerhalb des zulässigen Bereichs und das Minimum innerhalb. Bezogen auf den zulässigen Bereich ist dieses Minimum ein globales. Die Formeln für diese Extremstellen sind in (32) dargestellt. Die Formeln sind zwar komplex, jedoch sind die Ergebnisse innerhalb des zulässigen Bereichs reell.

$$z_1 = \frac{h\cdot(b-2\cdot y)}{4\cdot(b-y)} + \sqrt[3]{D+E} + \frac{F}{\sqrt[3]{D+b_1}}$$

$$z_2 = 0.5\cdot\left[\frac{h\cdot(b-2\cdot y)}{2\cdot(b-y)} - \sqrt[3]{D+E} - \frac{F}{\sqrt[3]{D+E}} + \sqrt{3}\cdot\left(\sqrt[3]{D+E} - \frac{F}{\sqrt[3]{D+E}}\right)\cdot i\right]$$

$$z_3 = 0.5\cdot\left[\frac{h\cdot(b-2\cdot y)}{2\cdot(b-y)} - \sqrt[3]{D+E} - \frac{F}{\sqrt[3]{D+E}} - \sqrt{3}\cdot\left(\sqrt[3]{D+E} - \frac{F}{\sqrt[3]{D+E}}\right)\cdot i\right] \qquad (32)$$

$$D = \frac{h^3\cdot y}{16\cdot(b-y)}\cdot\sqrt{\left[1 + \frac{2\cdot y^2 - 6\cdot b\cdot y - \frac{b^3}{y} + 4.5\cdot b^2}{2\cdot(b-y)^2} + \frac{2\cdot y - 3\cdot b}{b-y}\right]}$$

$$E = \frac{h^3}{32\cdot(b-y)}\cdot\left[\frac{b^3 - 8\cdot y^3 + 12\cdot b\cdot y^2 - 6\cdot b^2\cdot y}{2\cdot(b-y)^2} + \frac{3\cdot b\cdot y - 6\cdot y^2}{b-y} - 2\cdot y\right]$$

$$F = \frac{h^2}{4\cdot(b-y)}\cdot\left[\frac{(y-0.5\cdot b)^2}{b-y} + y\right]$$

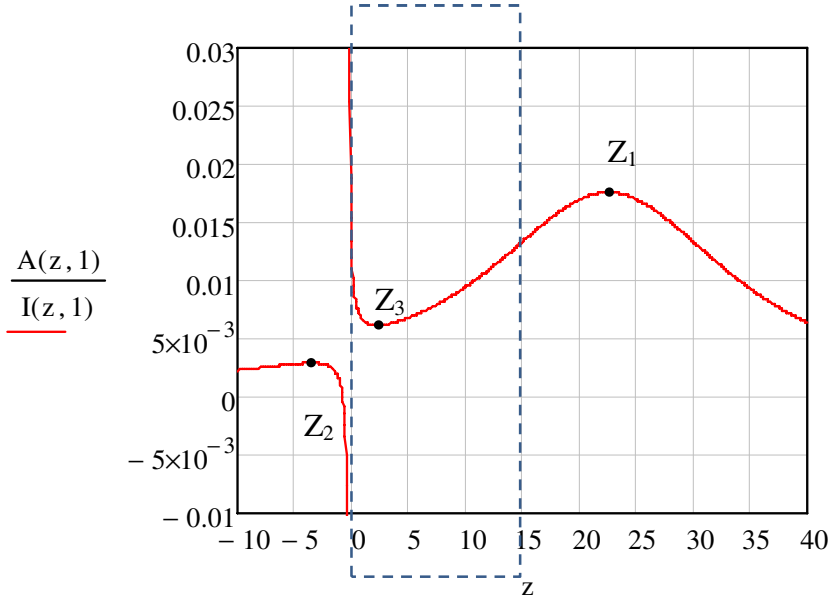


Abbildung 4.3: Die drei Extremstellen der Zielfunktion F(z)

In Abbildung 4.3 sind die drei Extremstellen und die Polstelle für die Abmessungen $h = 30\,mm, b = 20\,mm$ und $y = 1\,mm$ graphisch dargestellt. Der zulässige Bereich für z mit $0 \leq z \leq h/2$ ist strichliert eingezeichnet. Das globale Minimum des zulässigen Bereichs liegt bei $z_3 = 2{,}26\,mm$. Außerhalb des zulässigen Bereich befinden sich das erste lokale Maximum, liegend bei z_2 = -3,487 mm, die Polstelle bei $-0{,}789\,mm$ und das zweite lokale Maximum liegend bei $z_1 = 22{,}542\,mm$.

4.1.3 Finite-Elemente-Modell

Die Finite-Elemente-Berechnung wird mit dem Programm Abaqus durchgeführt. Das Netz ist so aufgebaut, dass es bei Änderungen der Querschnittsabmessungen des I-Trägers mit skaliert wird. Die Modellgeometrie, die Netzgeometrie und die Randbedingungen sind in der Textdatei *Traeger.inp* festgelegt. Der Träger besteht aus 3D Elementen, welche nur translatorische Freiheitsgrade besitzen. Dadurch kann an ihnen kein Moment angreifen. Deshalb wird, wie in Abbildung 4.4 dargestellt, am Ende des Trägers noch ein aus Balkenelementen aufgebauter Teil angebracht, an welchem ein Moment angreifen kann. In Abbildung 4.4 sind die Variablen blau und die Konstanten schwarz bemaßt.

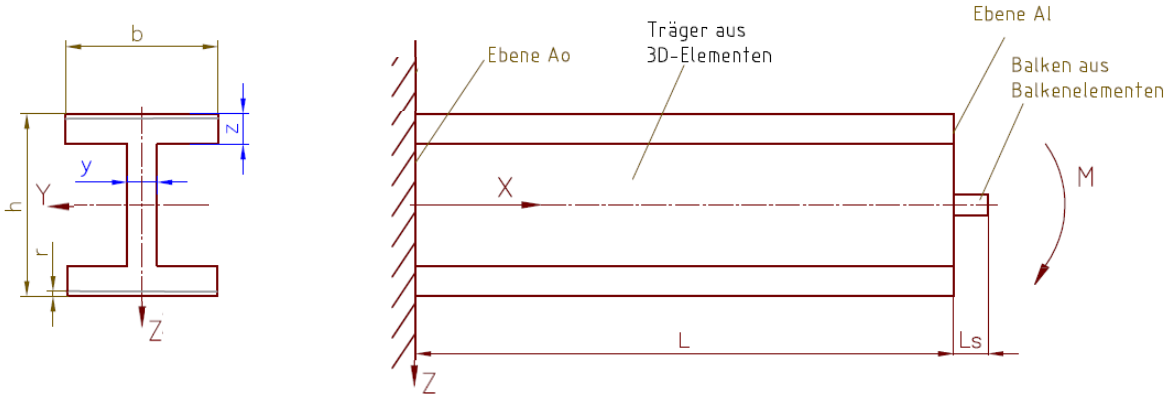


Abbildung 4.4: Skizze des Finite-Elemente-Modells für den I-Träger

Für die Zielfunktionsberechnung wird die maximale Durchbiegung $w_{z\,max}$, welche in z-Richtung auftritt, bzw. die maximalen Spannungen $\sigma_{x\,max}$ in x-Richtung benötigt. Es ist zu erwarten, dass die maximalen Spannungen in den äußersten Elementen der Gurte auftreten, wobei es sich um Normalspannungen σ_x in die x-Richtung handelt. Da die Spannungswerte an den Integrationspunkt am genausten berechnet werden, werden diese Werte verwendet. Um die Spannungswerte trotz Veränderung der Finite-Elemente-Netz immer an der gleichen Position auszuwerten, wird die Konstante r eingeführt, welche die Höhe dieser äußeren Elemente vorgibt, siehe Abbildung 4.4. Damit ist im Unterschied zu allen anderen Elementen, die Elementhöhe und dadurch auch die Lage der Integrationspunkte dieser Elemente unabhängig von der Gurtdicke. Dadurch können die Ergebnisse von Modellen mit unterschiedlicher Gurtdicke besser verglichen werden und es ist einfacher die Ergebnisse von Abaqus mit der analytischen Lösung zu vergleichen. Unabhängig von den anderen Abmessungen hat r den Wert $0{,}2\,mm$. Aufgrund dieser fixen Elementgröße kann dieses Modell nur Gurtdicken größer als $0{,}2\,mm$ berechnen, andernfalls wäre die Höhe der restlichen Elemente des Gurtes Null. Die Textdatei, welche das Finiten-Elemente-Model vom I-Träger beinhaltet, ist im Anhang A.3 Abbildung A. 9 und Abbildung A. 10 zu finden.

Netzaufbau

Um die Ergebnisse der Elementtypen C3D8 und C3D8R zu vergleichen, wird je ein Modell mit den am Anfang von Kapitel 4.1 beschriebenen Abmessungen, mit KC3 Randbedingungen

des nachfolgenden Kapitels „Einspannung“ und mit einem „normalen“ Netz wie im Kapitel „Netzfeinheit“ beschrieben, verwendet. Damit die Lage der Integrationspunkte der maximalen Spannung von Abaqus mit dem Randfaserabstand der analytischen Lösung übereinstimmt, wird mit dem Abaqus Befehl COORD die genaue Lage dieser bestimmt. Wie in Tabelle 4.1 ersichtlich, unterscheiden sich die mit Abaqus ermittelten maximalen Spannungen bei beiden Elementtypen nur minimal von der analytischen Lösung. Da bei der Berechnung mit den C3D8R Elementen der relative Fehler kleiner und die Rechenzeit etwas geringer ist, werden diese Elemente für weitere Berechnungen verwendet. Diese Elemente haben außerdem noch den Vorteil, dass ihr Integrationspunkt genau in der Mitte liegt. Dadurch kann man die Spannungen besonders leicht mit der analytischen Lösung vergleichen.

Tabelle 4.1: Vergleich Elemente mit reduzierter und voller Integration

	Randfaserabstand	$\sigma_{x\ max}$		$w_{z\ max}$		Zeit in s
		Wert in N/mm^2	relativer Fehler in %	Wert in mm	relativer Fehler in %	
analytische Lösung	h/2 - 0,044 mm	544,430	-	2,620	-	-
	h/2 - 0,1 mm	542,328	-		-	-
C3D8	h/2 - 0,044 mm	543,860	0,105	2,619	0,047	1,89
C3D8R	h/2 - 0,1 mm	542,427	0,018	2,620	0,014	1,86

Material

Im 3D Modell wird für den Werkstoff Aluminium ein elastisches isotropes Material verwendet, der E-Module wird mit $70000\ N/mm^2$ und die Poissonzahl mit $0{,}33$ gewählt. Für die Berechnung der Masse wird die Dichte des Aluminiums mit $2700\ kg/m^3$ angegeben.

Vergleich Abaqus mit analytischer Lösung

Für den Vergleich mit dem Abaqus-Ergebnis wird die mit der Formel (24) berechnete maximale Durchbiegung verwendet, welche am oberen Ende des Trägers auftritt. Um die Spannungen von Abaqus mit den analytisch ermittelten vergleichen zu können, muss der Randfaserabstand von der analytisch ermittelten Spannung mit der Lage des Integrationspunktes übereinstimmen. Bei den verwendeten Elementen C3D8R liegt der Integrationspunkt in der Elements Mitte. Somit ergibt sich für die maximale Spannung mit der am Anfang von Kapitel 4.1.3 eingeführten Variable r ein Randfaserabstand

$$e = \frac{h - r}{2} \tag{33}$$

.

Die Formel für die maximale Spannung lautet demnach:

$$\sigma_{x\,max} = \frac{M}{I_y}\frac{h - r}{2} \tag{34}$$

Netzfeinheit

Um zu überprüfen wie sich das Ergebnis verhält, wenn das Netz feiner wird, werden drei unterschiedlich feine Netze erstellt, die mit „feines“, „normales“ und „grobes“ Netz bezeichnet werden. Variiert wird die Elementanzahl über die Stegdicke y, die Steghöhe h_s, die Gurtbreite b, die Gurtdicke z und über die Trägerlänge l, siehe
Abbildung 4.5 und Tabelle 4.2. Es werden für den Vergleich dieser Netze, die am Anfang von Kapitel 4.1 beschriebenen Abmessungen und die Randbedingungen KC3 des nachfolgenden Kapitels „Einspannung“ verwendet.

Tabelle 4.2: Anzahl der Elemente

	feines Netz	normales Netz	grobes Netz
Gurtbreite b	56	36	16
Gurtdicke z	7	5	5
Stegdicke y	6	6	6
Steghöhe h_s	34	28	8
Trägerlänge l	120	100	100

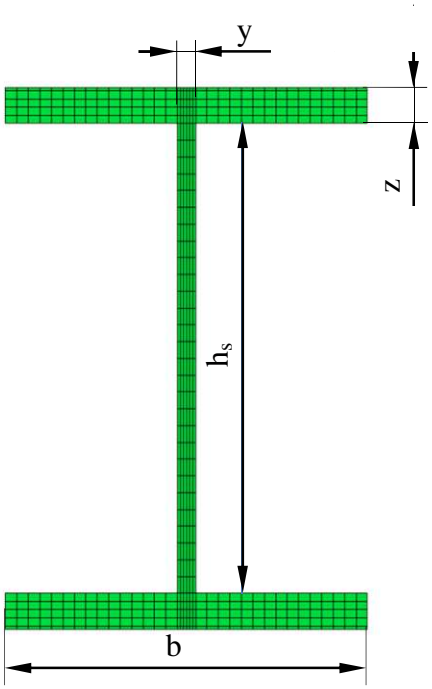


Abbildung 4.5: Anzahl der Elemente

Alle drei Modelle zeigen den gleichen erwarteten Spannungs- und Verformungsverlauf, in Abbildung 4.6 und Abbildung 4.7 sind die Ergebnisse vom „normalen“ Netz dargestellt. Die Normalspannungen σ_x sind linear über die Trägerhöhe verteilt und die maximalen Werte treten in den äußeren Gurtelementen auf. Die maximalen Verschiebungen w_z treten am freien Ende des Trägers auf. Sowohl bei den Verschiebungen w_z, als auch bei den Normal-

spannungen σ_x unterscheiden sich bei den drei Modellen die maximalen Werte um weniger als 1% bezogen auf die analytische Lösung.

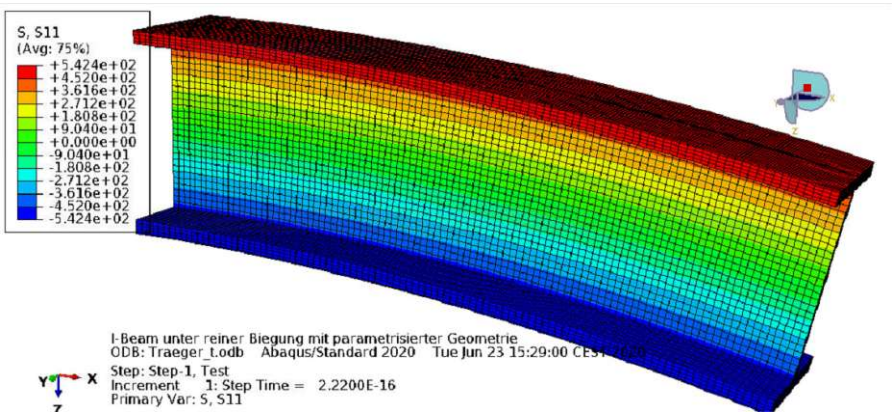


Abbildung 4.6: Normallspannungsverlauf $\sigma_x = S_{11}$ in N/mm^2 beim „normalen“ Netz

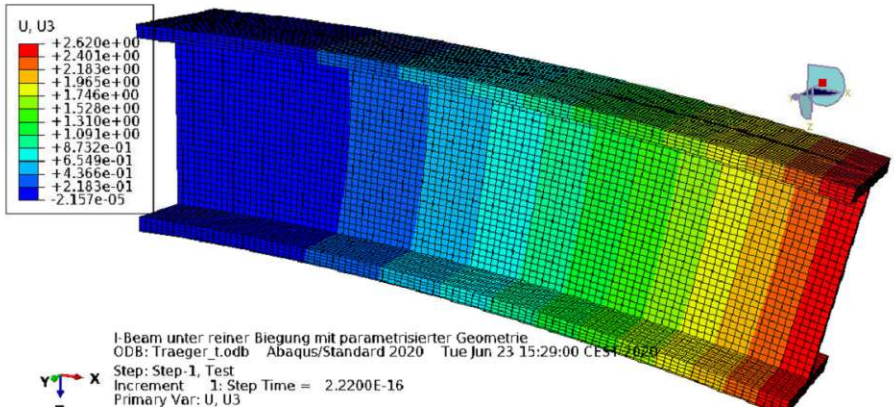
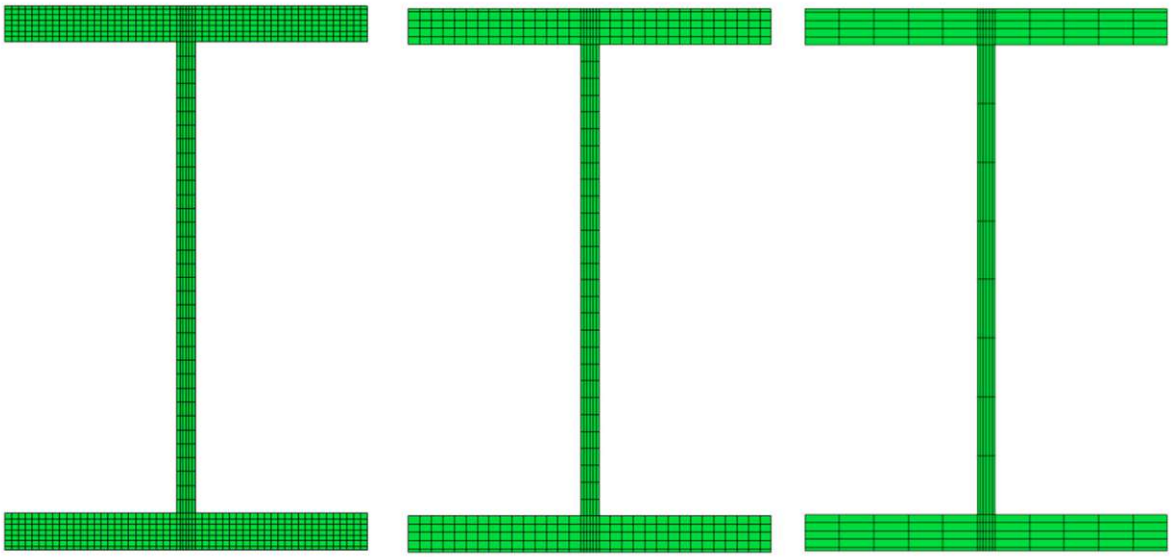


Abbildung 4.7: Verschiebungen $w_z = U_3$ in mm beim „normalen“ Netz

In Abbildung 4.8 sind die drei unterschiedlichen Netze dargestellt. Das „normale“ Netz besteht aus 52803 Elementen und 64573 Knoten. Die Elementanzahl entlang der Gurtdicke beträgt 5, entlang der Stegdicke 6, entlang der Gurtbreite 36, entlang der Steghöhe 28 und entlang der Trägerlänge 100 Elemente. Abaqus hat bei einer Rechenzeit von 1,86 s eine maximale Spannung $\sigma_{x\,max}$ von $542{,}43\ N/mm^2$ und eine maximale Durchbiegung $w_{z\,max}$ von $2{,}6196\ mm$ ermittelt.

Das „grobe“ Netz hat nicht einmal halb so viele Elemente bzw. Knoten wie das „normale“ Netz, nämlich nur 20803 Elemente und 25813 Knoten. Die Anzahl der Elemente über die Gurtbreite wurde auf 16 verringert und über die Steghöhe auf 8. Dafür ist die Berechnung um mehr als die Hälfte schneller, es werden nur $0{,}83\ s$ benötigt. Die maximale Spannung $\sigma_{x\,max}$ beträgt $543{,}09\ N/mm^2$ und die maximale Durchbiegung $w_{z\,max}$ hat einen Wert von $2{,}6227\ mm$.



„feines“ Netz „normales“ Netz „grobes“ Netz

Abbildung 4.8: Gegenüberstellung von „feinen“, „normalen“ und „groben“ Netz

Tabelle 4.3: Vergleich der unterschiedlich feinen Netze

Netz	Anzahl		Zeit in s	Spannung $\sigma_{x\,max}$		Durchbiegung $w_{z\,max}$	
	Elemente	Knoten		Wert in N/mm^2	relativer Fehler in %	Wert in mm	relativer Fehler in %
analytische Lösung				542,33	-	2,62	-
"grob"	20803	25813	0,83	543,09	0,14	2,62	0,11
"normal"	52803	64573	1,86	542,43	0,02	2,62	0,01
"fein"	118563	139453	4,42	542,39	0,01	2,62	0,02

Um die Ergebnisse bewerten zu können, wurde der relative Fehler, bezogen auf die analytische Lösung für die Spannung bzw. für die Durchbiegung, ermittelt. In Tabelle 4.3 ist klar ersichtlich, dass die reltiven Fehler vom „normalen“ Netz kleiner sind als jene, des „groben“. Das „normale“ Netz liefert also genauere Ergebnisse, jedoch bei einer längeren Rechenzeit.

Obwohl das „feine“ Netz mehr als doppelt so viele Elemente bzw. Knoten als das „normale“ Netz besitzt, sind die Ergebnisse nicht genauer, aber die Berechnung dauert mit 4,42 s mehr als doppelt so lange. Mit 36 Elementen entlang der Gurtbreite, 6 entlang der Stegdicke, 34 entlang der Steghöhe, 120 entlang der Länge und 7 entlang der Gurtdicke, hat dieses Modell 118563 Elemente und 139453 Knoten. Die Spannung $\sigma_{x\,max}$ beträgt 542,39 N/mm^2 und die Durchbiegung $w_{z\,max}$ 2,6194 mm. Aufgrund dieser Erkenntnisse wurde für die Optimierung das „normale“ Netz verwendet. Das Finite-Elemente-Model des I-Trägers, welches im Anhang Abbildung A. 10 zu finden ist, beruht auf dem „normalen“ Netz.

Einspannung

Am eingespannten Ende des Trägers, hat die erste Knotenebene den Namen A_0, siehe Abbildung 4.4. Bei jedem Knoten der Ebene A_0 ist die Verschiebung in die x-Richtung gesperrt. Dadurch wird eine Starrkörperverschiebung in x-Richtung, die Starrkörperrotation

um die z-Achse und die Starrkörperrotation um die y-Ache unterbunden. Bei zwei Knoten von der A_0 Ebene sind auch noch weitere Freiheitsgrade gesperrt, siehe Abbildung 4.9. Beim „normalen“ Netz sind beim Knoten 740 in der Mitte des Profils auch noch die zwei restlichen Freiheitsgrade, also Verschiebung in y-Richtung und z-Richtung, gesperrt. Dadurch wird eine Starrkörperverschiebung des Trägers in die z- und y-Richtung verhindert. Beim Knoten 721, welcher am unteren Gurtende auf der z-Achse liegt, ist noch zusätzlich die Verschiebung in y-Richtung gesperrt. Dadurch kann keine Starrkörperrotation um die x-Achse erfolgen und der Körper ist im Raum fixiert.

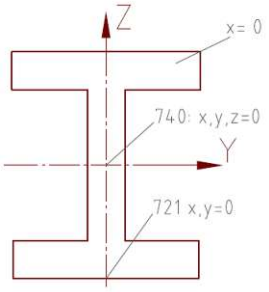


Abbildung 4.9: Ebene A_0 mit Randbedingungen beim „normalen“ Netz

Kopplung Balken Träger

Um den Übergang von den 3D Elementen zu den Strukturelementen nicht zu steif auszuführen, wurden drei unterschiedliche Verbindungen ausprobiert. Für den Vergleich der unterschiedlichen Randbedingungen wurde das „normale“ Netz verwendet und die am Anfang von Kapitel 4.1 beschriebenen Abmessungen.

An jenem Ende des Trägers, an dem das Moment angreift und auch die Balkenelemente sich befindet, heißt die erste Knotenebene der 3D-Elemente A_1 Ebene, siehe Abbildung 4.4. Der Befehl Kinematic Coupling (KC), welcher für die Kopplung von Träger und Balken verwendet wird, erzeugt eine steife Verbindung zwischen dem ausgewählten Knotenset und dem Referenzknoten. Der Befehl bewirkt, dass die ausgewählten Freiheitsgarde des angegeben Knotenset von der Bewegung des angegeben Referenzknoten abhängen. Wenn zum Beispiel beim Knotenset der Freiheitsgrad x ausgewählt ist, dann bleibt der relative Abstand in x-Richtung zwischen den Set und den Referenzknoten konstant.

Beim Modell KC1 sind alle Freiheitsgarde aller Knoten der A_1 Ebene beim Befehl Kinematic Coupling (KC) ausgewählt. Der Referenzknoten ist der erste Knoten des Balkens. Diese Verbindung ist zu steif und es kommt zu Randstörungen im Bereich der Verbindung, wie in Abbildung 4.10 ersichtlich.

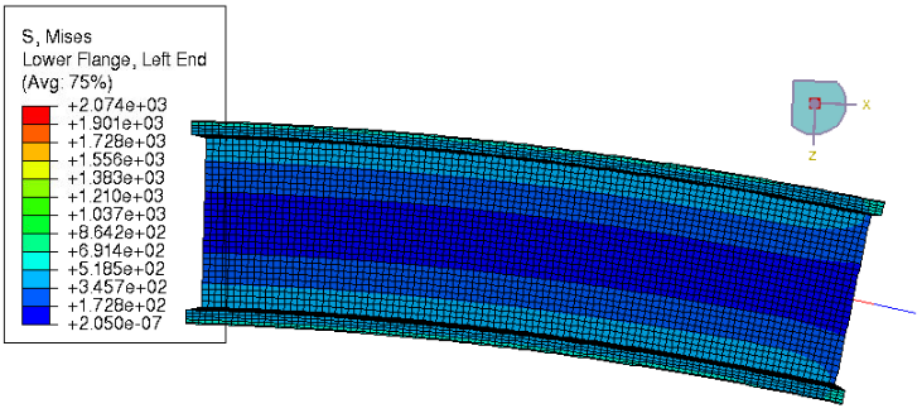


Abbildung 4.10: Modell KC1, Mises-Vergleichsspannung in N/mm^2

Beim Modell KC2 sind nur die Freiheitsgrade x, φ_x, $\varphi_{y,}$ und φ_z aller Knoten der A_l Ebene beim Befehl Kinematic Coupling (KC) ausgewählt. Der Referenzknoten ist der erste Knoten des Balkens. Es kommt zu keinen Randstörungen mehr, jedoch wird die Lage des Balkens nicht richtig dargestellt, siehe Abbildung 4.11.

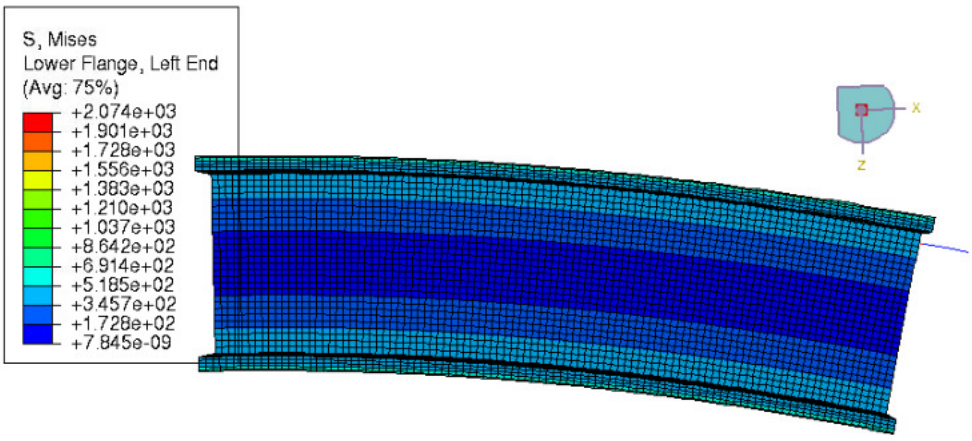


Abbildung 4.11: Modell KC2, Mises-Vergleichsspannung in N/mm^2

Beim Modell KC3 wird der Befehl Kinematic Coupling zweimal angewendet. Beim ersten Mal sind nur die Freiheitsgrade x, φ_x, $\varphi_{y,}$ und φ_z aller Knoten der A_l Ebene ausgewählt. Das zweite Mal wird nur der Knoten in der Mitte der A_l Ebene und die Freiheitsgrade z und y verwendet. Der Referenzknoten ist beide Male der erste Knoten des Balkens. Wie in Abbildung 4.12 ersichtlich, treten bei diesem Modell die gleichen Verschiebungen und Spannungen auf wie beim Modell KC2 und die Lage des Balkens wird richtig dargestellt.

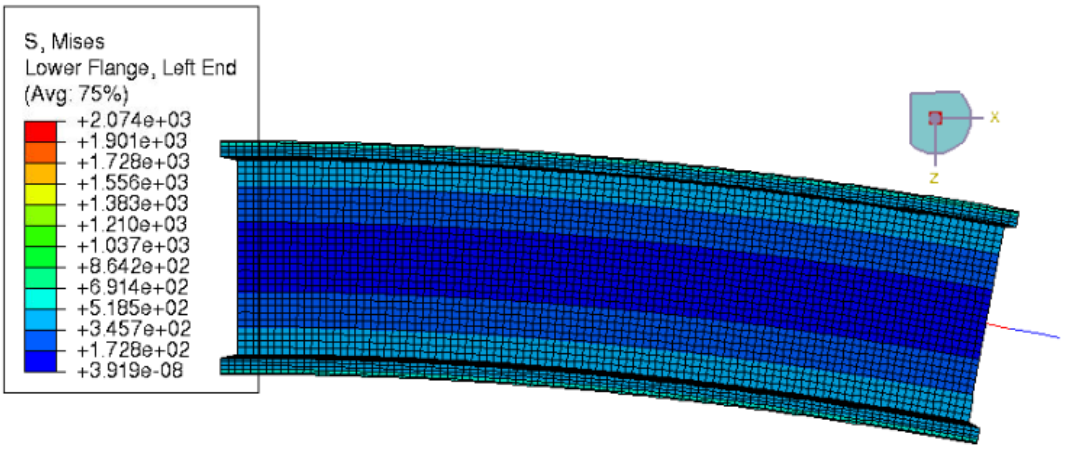


Abbildung 4.12: Modell KC3, Mises-Vergleichsspannung in N/mm^2

In Tabelle 4.4 sind die Ergebnisse zusammengefasst. Für die Optimierungen wurde das Model KC3 verwendet, da dieses Modell sowohl die Spannungen als auch die Lage des Balkens richtig darstellt.

Tabelle 4.4: Vergleich unterschiedliche Rand Bedingungen

	Kopplung des ersten Balkenknoten mit		Spannung $\sigma_{x\,max}$		Durchbiegung $w_{z\,max}$		Richtige Grafische Darstellung
	KC						
	Knoten der AL Ebene	gekop-pelte DOF	Wert in N/mm^2	rel. Fehler in %	Wert in mm	rel. Fehler in %	
analy-tisch	-	-	542,3	-	2,62	-	-
KC 1	alle	x, y, z, φ_x, φ_y, φ_z	754,1	39	2,06	21	ja
KC 2	alle	x, φ_x, φ_y, φ_z	542,4	0,01	2,62	0,01	nein
KC 3	alle	x, φ_x, φ_y, φ_z	542,4	0,01	2,62	0,01	ja
	Mittel-knoten	y,z					

Limits

Die Grenzen des Abaqus Models wurden wie folgt festgelegt:

$$1\ mm < z < \frac{h - 1\ mm}{2} = 14{,}5\ mm$$

$$1\ mm < y < b - 1\ mm = 19\ mm$$

Um abschätzen zu können, wie groß die maximal auftretenden relativen Fehler des Modells gegenüber der analytischen Lösung sind, werden für die Kombinationen aus den maximalen und minimalen Werten von z und y die maximalen Spannungen $\sigma_{x\,max}$ und die maximalen Durchbiegungen $w_{z\,max}$ ermittelt. In Abbildung 4.13 sind diese Kombinationen grafisch dargestellt und in der Tabelle 4.5 sind die Werte zusammengefasst. Verwendet wird dafür das

„normale“ Netz mit den am Anfang von Kapitel 4.1 beschriebenen Abmessungen und Materialwerten.

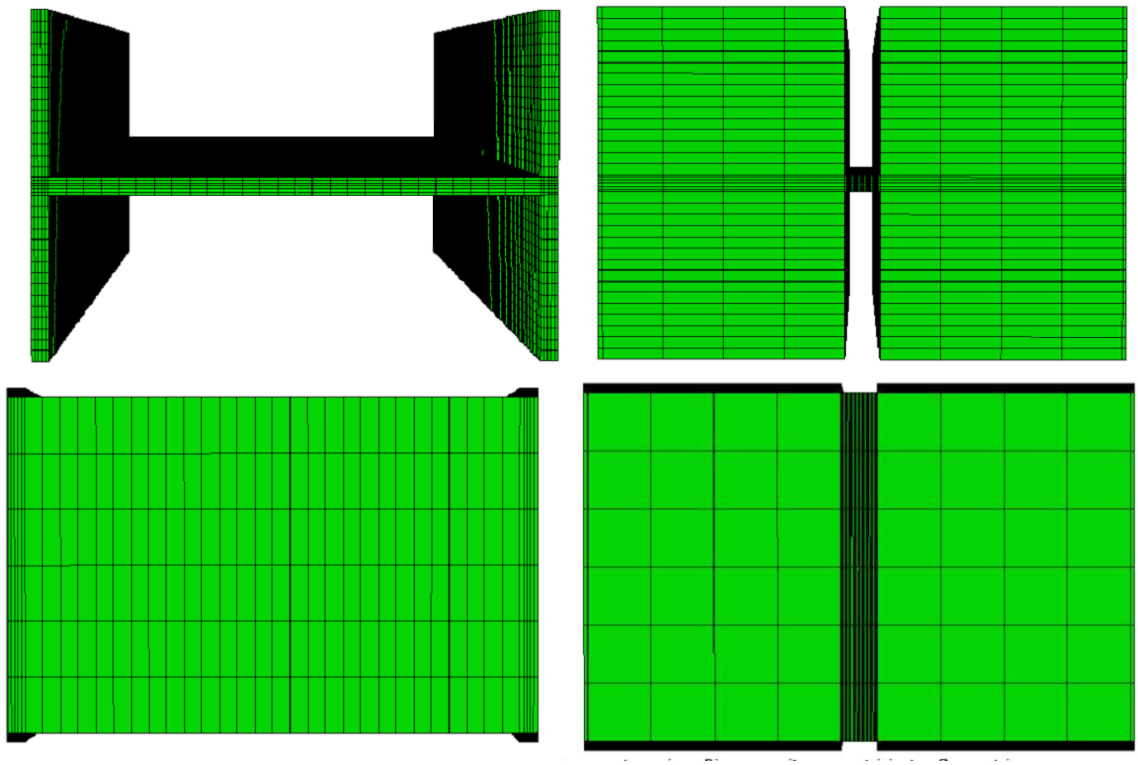


Abbildung 4.13: Darstellung der Grenzen

Tabelle 4.5: Relative Fehler an den Grenzen des Modells

z	y	Abaqus		analytisch		relativer Fehler in %	
		$\sigma_{x\,max}$	$w_{z\,max}$	$\sigma_{x\,max}$	$w_{z\,max}$	$\sigma_{x\,max}$	$w_{z\,max}$
1	1	909,405	4,392	909,187	4,415	0,024	0,523
14,5	1	209,779	1,013	206,952	0,995	1,366	1,811
1	19	215,935	1,043	215,714	1,037	0,102	0,563
14,5	19	209,771	1,013	206,945	0,995	1,366	1,807

Es ist vertretbar, dass die Abweichungen bei großer Gurtdicke z größer werden, da das Gewicht minimieren werden soll und es daher unwahrscheinlich ist, dass z maximal ist.

4.1.4 Optimierungsergebnisse

Um das best mögliche Ergebnis zu erhalten, wird das „normale“ Finite-Elemente-Netz verwendet. Die Ergebnisse von Abaqus werden mithilfe des odb-Files in Dakota eingelesen. Zur Anwendung kommen die zwei Zielfuntionen $\sigma_{x\,max} \cdot A$ und einmal $\sigma_{x\,max} \cdot A$. Wie erwartet, erhält man, unabhängig von den beiden Zielfunktionen, das gleiche Ergebnis.

Die geometrischen Abmessungen und die Optimierungsvariablen in diesem Kapitel sind gleich wie jene, für welche in Kabitel 4.1.2 eine analytische Lösung gefunden wurde. Es werden Optimerungen mit einer Variable und mit zwei Variablen durchgeführt. Wenn nur eine Variable verwendet wird, handelt es sich um die Gurtdicke z mit einem zulässigen Bereich von 1 mm bis 14,5 mm. Werden zwei Variablen untersucht, wird zusätzlich zur

Grutdicke auch noch die Stegdicke y mit einem zulässigen Bereich von 1 mm bis 19 mm verwendet. Ohne weitere Angabe, sind folgende Abmessungen y = 1 mm, h = 30 mm, l = 100 mm, r = 0,2 mm und b = 20 mm als konstant angenommen.
Die ersten vier Formoptimierungen „ Optimierungsaufgabe 1 - 4“ werden mit der Variable z durchgeführt. Bei allen ist das Ziel, die Durchbiegung, die Spannung und das Gewicht zu minimieren und zwar mit der Zielfunktion $w_{z\,max} \cdot A$ bzw. $\sigma_{x\,max} \cdot A$. Bei der ersten Optimierungsaufgabe gibt es keine Nebenbedingungen. Bei der zweiten kommt jene hinzu, dass die Durchbiegung einen maximalen Wert nicht überschreiten darf. Bei der dritten wird anstelle der Durchbiegung, die Spannung limitiert. Zum Schluss wird in der vierten Optimierungsaufgabe sowohl für die Durchbiegung als auch für die Spannung ein maximaler Wert vorgegeben.
Da die Formoptimierung mit der Variable z erfolgreich ist, wird versucht, das Gewicht, die Spannung und die Durchbiegung auch mit zwei Variablen zu optimieren. Die anderen geometrischen Größen und Materialkennwerte sind ident mit jenen der Optimierung für die Variable z. Es werden wieder vier unterschiedliche Optimierungen durchgeführt. Bei der „Optimierungsaufgabe 5“ gibt es keine Nebenbedingung. Bei der „Optimierungsaufgabe 6“ ist die Durchbiegung und bei der „Optimierungsaufgabe 7“ die Spannung limitiert und bei der Aufgabe 8 sind beide begrenzt.

Optimierungsaufgabe 1

Der optimale Wert z_{opti} für die Gurtdicke z soll ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind. Diese Optimierungsaufgabe ist gleich wie jene, für welche analytisch ein Optimum bei z_3 = 2,26 mm gefunden wurde. Die numerischen Optimierungsergebnisse z_{opti} für die Variable z sind in Tabelle 4.6 dargestellt. Bei zwei Optimierungen wurde die Zielfunktion $\sigma_{x\,max} \cdot A$ verwendet, bei den restlichen die Zielfunktion $w_{z\,max} \cdot A$. Der Optimieralgorithmus *coliny_direct* kommt bei beiden Zielfunktionen auf genau das gleiche Ergebnis für z_{opti} und beim Optimierer *coliny_cobyla* unterscheiden sich die Ergebnisse erst in der vier Nachkommerstelle.

Tabelle 4.6: Ergebnis der „Optimierungsaufgabe 1“

Optimierer von Dakota	Optimierungsergebnis			relativer Fehler in %		Zeit
	z_{opti} in mm	Zielfunktion in mm^3 bzw N		z_{opti}	Ziel-funktion	
coliny_cobyla	2,259	$w_{z\,max}$·A	277,150	0,06	0,00	6 min
coliny_direct	2,259	$w_{z\,max}$ ·A	277,150	0,05	0,00	1 h 3 min
coliny_ea	2,258	$w_{z\,max}$ ·A	277,150	0,09	0,00	4 h 57 min
coliny_ea	2,263	$w_{z\,max}$ ·A	277,151	0,13	0,00	5 h 55 min
efficient_global	2,259	$w_{z\,max}$ ·A	277,150	0,05	0,00	5 min
efficient_global	2,259	$w_{z\,max}$ ·A	277,150	0,04	0,00	5 min
ncsu_direct	2,259	$w_{z\,max}$ ·A	277,150	0,05	0,00	5 h 39 min
soga	2,256	$w_{z\,max}$ ·A	277,151	0,20	0,00	2 h 32 min
soga	2,272	$w_{z\,max}$ ·A	277,152	0,54	0,00	4 h 39 min
coliny_cobyla	2,259	$w_{z\,max}$ ·A	57387,7	0,07	0,00	10 min
coliny_direct	2,259	$\sigma_{x\,max}$ ·A	57387,7	0,05	0,00	54 min

Alle sechs verwendeten Optimierungsalgorithmen liefern für diese Aufgabenstellung sehr gute Ergebnisse, die relativen Fehler sind kleiner als 1%. Für die Berechnung des relativen Fehlers von z_{opti} wurde das Ergebnis von Dakota mit der analytischen Lösung von Kapitel 4.1.2 verglichen. Um einen Referenzwert für die Berechnung des relativen Fehlers von der Zielfunktion zu haben, wurde eine Abaqus-Berechnung durchgeführt, bei welchen z den Wert z_3 von der analytischen Optimierung hat. Für die Zielfunktion $w_{z\,max} \cdot A$ wurde hiermit der Referenzwert 275,056 mm^3 ermittelt und für $\sigma_{x\,max} \cdot A$ der Wert 57389,9 N/mm^2.
Auffallend ist, dass *efficient_global* und *coliny_cobyla* das Optimum am schnellsten gefunden haben. Bei den deterministischen Algorithmen *coliny_direct, coliny_cobyla* und *ncsu_direct* ist jeweils nur ein Optimierungsergebnis angeführt. Denn auch, wenn man mit einem von ihnen mehrere Optimierungen durchführt, kommt man, wie erwartet, immer zum gleichen Ergebnis. Die restlichen drei verwendeten Algorithmen beruhen auf stochastische Verfahren, das erkennt man gut daran, dass sich die Ergebnisse eines Algorithmus minimal voneinander unterscheiden.
In Abbildung 4.14 sind die möglichen Zielfunktionen über den zulässigen Bereich von z dargestellt. Es ist gut zu erkennen, dass sich sowohl $w_{z\,max} \cdot A$ als auch $\sigma_{x\,max} \cdot A$ und A/I_y als Zielfunktion eignen, da alle drei den gleichen Kurvenverlauf haben und sich daher bei der Optimierung gleich verhalten. Die Zielfunktionen $w_{z\,max} \cdot A$ und $\sigma_{x\,max} \cdot A$ wurden verwendet, wenn das Abaqus-Modell der Optimierung zugrunde liegt und A/I_y zur Kontrolle, wenn anstelle dieses Finite-Elemente-Modells das Flächenträgheitsmoment analytisch hinterlegt ist. Der Zielfunktionswert für das Optimum ist blau punktiert dargestellt.

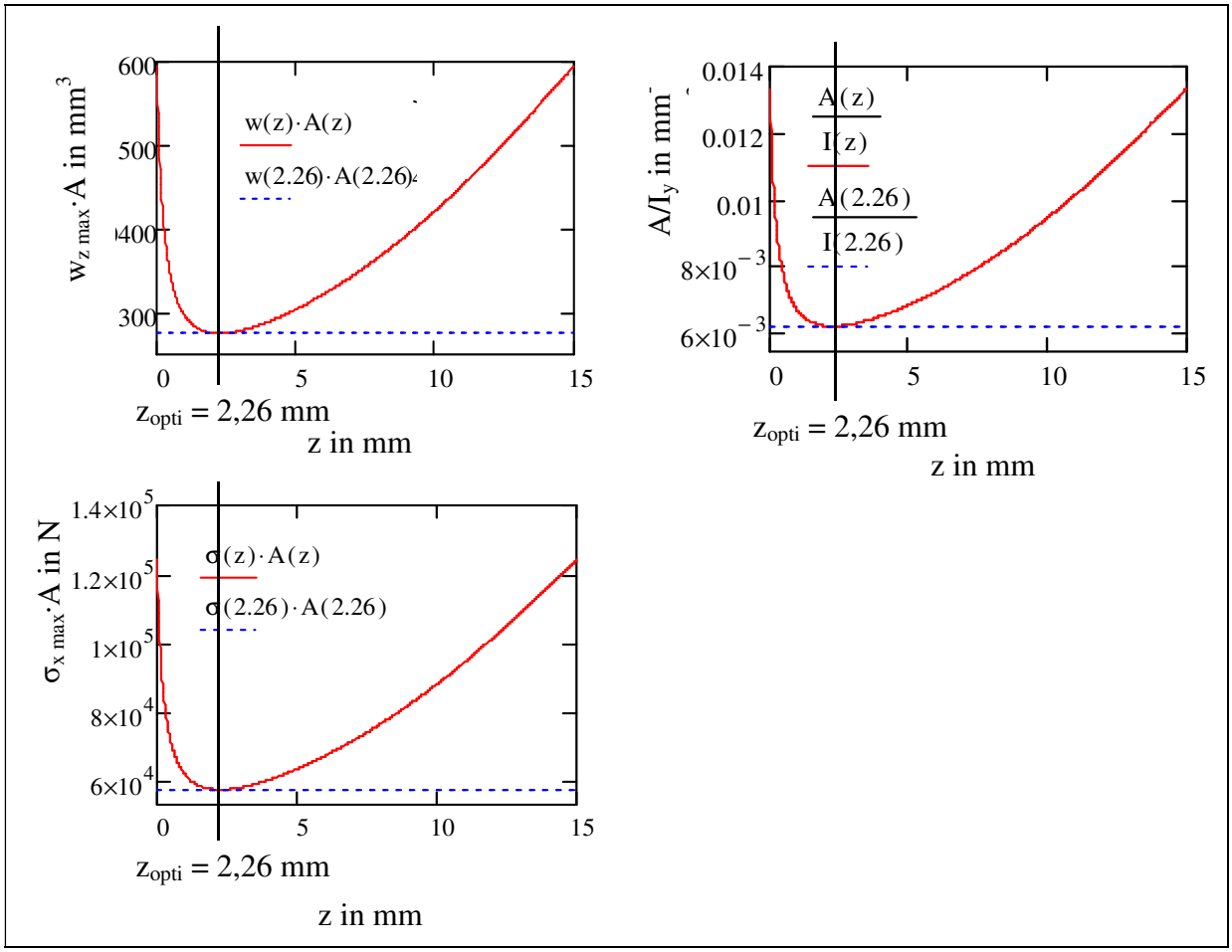


Abbildung 4.14: Graphische Darstellung der Zielfunktionen

Optimierungsaufgabe 2, Durchbiegung limitiert

Der optimale Wert z_{opti} für die Gurtdicke z soll ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind und für die Durchbiegung ist auch ein konkreter Wert gegeben, welcher nicht überschritten werden darf. In der Spalte Limit findet man den jeweiligen Grenzwert für die Durchbiegung. In Abbildung 4.16 ist sowohl der Verlauf der Durchbiegung, als auch jener von der Zielfunktion $w_{z\,max} \cdot A$ über die Optimierungsvariable z dargestellt. Die Grenze, welche sich auf Grund der Restriktion $w_z \leq 2$ mm ergibt, ist orange eingezeichnet. Es ist gut zu erkennen, dass das Minimum der Zielfunktion von z = 2,26 mm links vom Grenzwert und damit außerhalb des zulässigen Bereichs liegt. Deshalb befindet sich das Optimum z_{opti} genau beim maximalen Wert der Durchbiegung. In der Tabelle 4.7 sind jene Felder grün gekennzeichnet, bei welchen das Ergebnis des Optimierers und die grafische bzw. analytische Lösung ident sind. Demnach haben alle Optimierer außer *coliny_direct* und *coliny_ea* das Optimum gefunden. *Coliny_cobyla* und *efficient_global* waren wieder mit Abstand am schnellsten.

Tabelle 4.7: Ergebnis der „Optimierungsaufgabe 2“

Optimierer von Dakota	z_{opti} in mm	$w_{z\,max}\cdot A$ in mm^3	$w_{z\,max}$ in mm		Zeit
			Ergebnis	Limit	
coliny_cobyla	2,95	279,91	1,97	≤ 1.97	4 min
coliny_cobyla	2,89	279,48	2,00	≤ 2	4 min
coliny_direct	3,00	280,29	1,95	≤ 1.97	1 h 2 min
coliny_direct	3,00	280,29	1,95	≤ 2	2 h 18 min
coliny_ea	3,36	283,52	1,80	≤ 1.97	13 h 19 min
coliny_ea	2,90	279,55	1,99	≤ 2	5 h 34 min
efficient_global	2,95	279,91	1,97	≤ 1.97	5 min
efficient_global	2,89	279,48	2,00	≤ 2	5 min
efficient_global	2,89	279,46	2,00	≤ 2	5 min
soga	2,95	279,93	1,97	≤ 1.97	7 h 14 min
soga	2,916	279,67	1,99	≤ 2	3 h 10 min

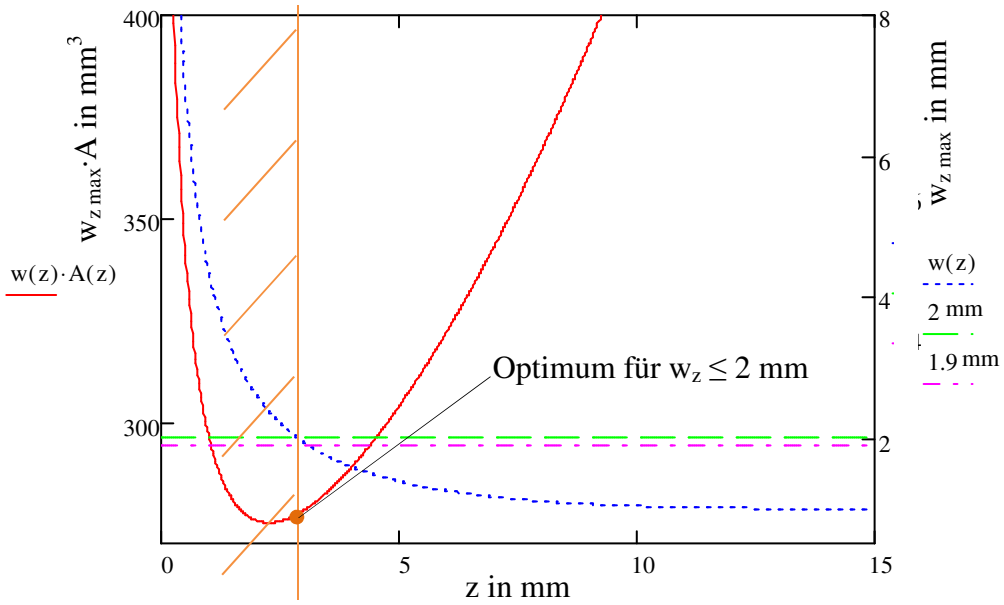


Abbildung 4.15: Grafische Darstellung der „Optimierungsaufgebe 2“ mit der Restriktion $w_z \leq 2$ mm

Optimierungsaufgabe 3, Spannung limitiert

Der optimale Wert z_{opti} für die Gurtdicke z soll ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind und für die Spannung ist auch noch ein konkreter Wert gegeben, welcher nicht überschritten werden darf. Da bei der zweiten Optimierungsaufgabe die Optimieralgorithmen *coliny_cobyla* und *efficient_global* gut und schnell konvergieren, werden bei der jetzigen Aufgabe nur mehr diese zwei verwendet. In Abbildung 4.16 ist der Verlauf der Zielfunktion und jener der Spannung abgebildet. Die horizontalen Linien stellen die Nebenbedingung $\sigma_x \leq 450$ N/mm^2 bzw. $\sigma_x \leq 400$ N/mm^2 dar.

Die blaue vertikale Linie kennzeichnet den zulässigen Bereich für $\sigma_{x\,max} \leq 450$ N/mm^2. Das Minimum der Zielfunktion z = 2,26 mm liegt in beiden Fällen außerhalb dieses Bereiches und das Optimum z_{opti} für diese Aufgabe liegt daher bei der maximal erlaubten Spannung. Die Optimierungen mit Dakota kommen zu demselben Ergebnis, wie in Tabelle 4.8 nachzulesen ist.

Tabelle 4.8: Ergebnis der „Optimierungsaufgabe 3“

Optimierer von Dakota	z_{opti} in mm	$w_{z\,max} \cdot A$ in mm^3	$\sigma_{x\,max}$ in N/mm^2		Zeit in min
			Ergebnis	Limit	
coliny_cobyla	3,03	280,54	400,00	≤ 400	3
coliny_cobyla	2,57	277,79	450,00	≤ 450	3
efficient_global	3,03	280,54	400,00	≤ 400	3
efficient_global	2,57	277,79	450,00	≤ 450	3

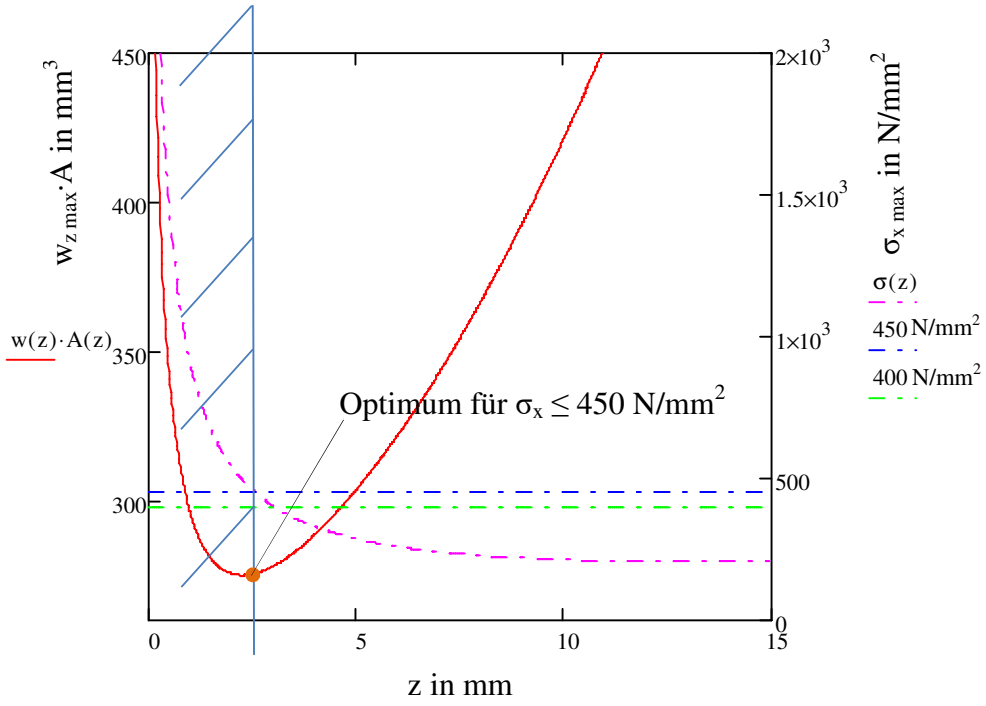


Abbildung 4.16: Grafische Darstellung der „Optimierungsaufgebe 3“ mit der Restriktion $\sigma_x \leq 450$ N/mm^2

Optimierungsaufgabe 4, σ_x und w_z limitiert

Der optimale Wert z_{opti} für die Gurtdicke z soll ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind. Zusätzlich sind für die Spannung und die Durchbiegung auch konkrete Werte gegeben, welche nicht überschritten werden dürfen. Die Restriktion für die Durchbiegung mit $w_z \leq 2\,mm$, in Abbildung 4.17 durch eine horizontale blaue Linie dargestellt, ist strenger als jene für die Spannung mit $\sigma_x \leq 450\,N/mm^2$, welche schwarz eingezeichnet ist. Daher und weil das Minimum außerhalb des zulässigen Bereiches ist, liegt das Optimum bei $w_{z\,max} = 2\,mm$. Die

Ergebnisse von dieser Optimierung sind in Tabelle 4.9 angeführt. Wenn man die Ergebnisse für z_{opti} von der „Optimierungsaufgabe 2“, mit dem Limit $w_z \leq 2\,mm$, mit den jetzigen Ergebnissen vergleicht, unterscheiden sie sich beim Optimieralgorithmus *coliny_cobyla* erst in der sechsten Nachkommastelle und bei *efficient_global* in der fünften Nachkommastelle.

Tabelle 4.9: Ergebnis der „Optimierungsaufgabe 2“

Optimierer	z_{opti} in mm	Zielfunktion in mm^3 bzw. N		$w_{z\,max}$ in mm	$\sigma_{x\,max}$ in N/mm^2	Grenze in mm bzw N/mm^2	Zeit in min
efficient_global	2,89	279,48	$w_{z\,max}\cdot A$	2,00	414,13	$w_z \leq 2$, $\sigma_x \leq 450$	4
coliny_cobyla	2,89	57870	$\sigma_{x\,max}\cdot A$	2,00	414,13	$w_z \leq 2$, $\sigma_x \leq 450$	3

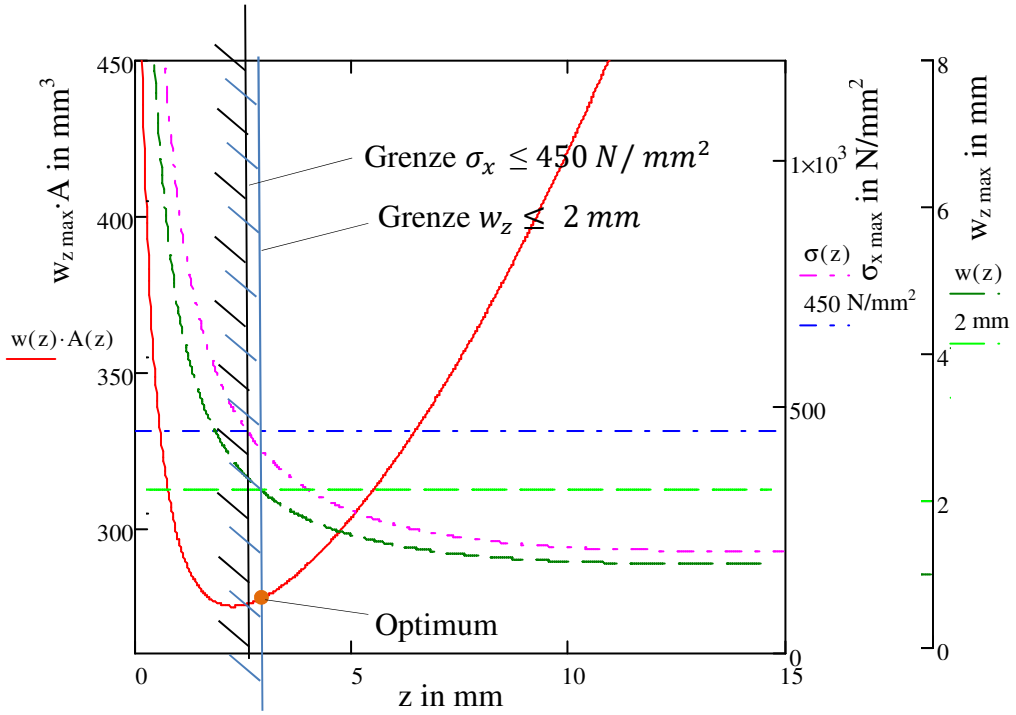


Abbildung 4.17: Grafische Darstellung der „Optimierungsaufgabe 4“

Optimierungsaufgabe 5

Der optimale Wert z_{opti} für die Gurtdicke z und y_{opti} für die Stegdicke y sollen ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind. Bei der ersten Optimierung mit den Variablen z und y, wird die Zielfunktion $w_{z\,max} \cdot A$ verwendet. Es gibt zwar keine analytische Lösung für diese Optimierungsaufgabe, jedoch kann man im 2D-Diagramm und im 3D-Diagramm gut erkennen, wo sich innerhalb des zulässigen Bereichs das Optimum z_{opti} befindet. Der zulässige Bereich von z und y ist wie folgt definiert: $1\text{ mm} < z < 14{,}5\text{ mm}$, $1\text{ mm} < y < 19\,mm$.

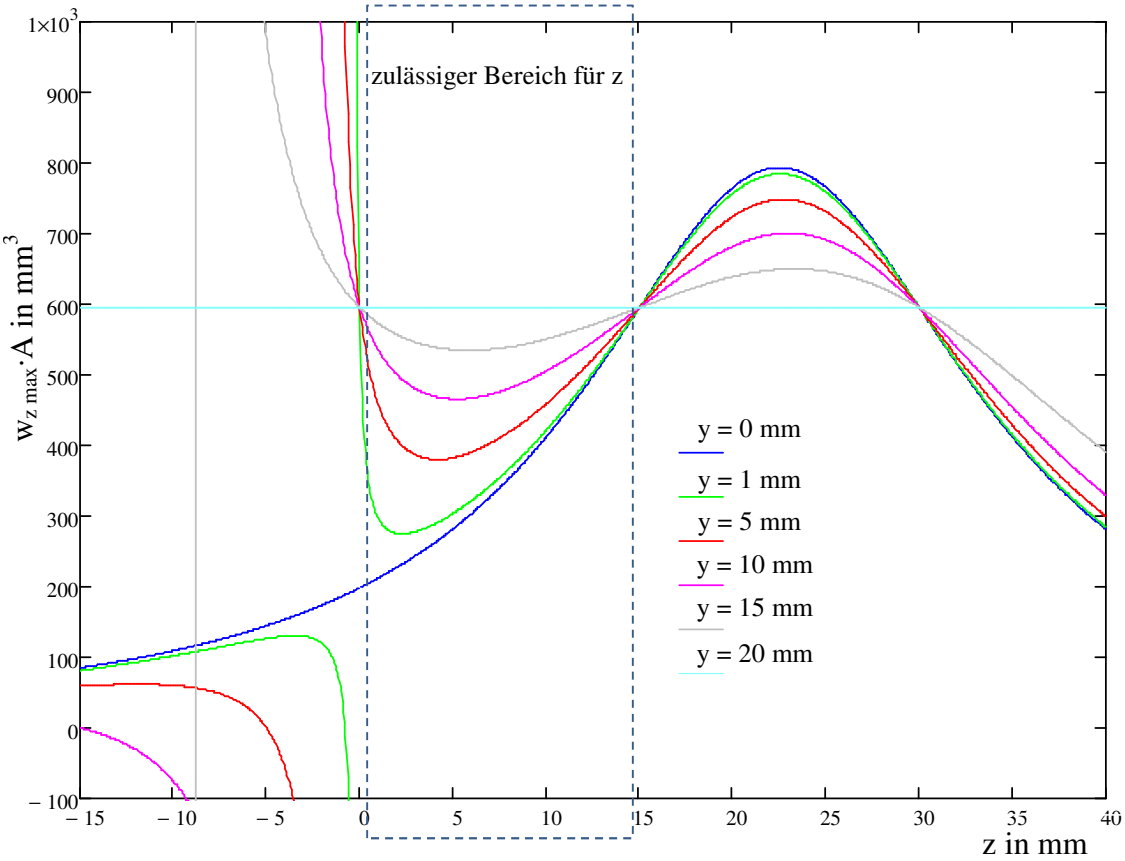


Abbildung 4.18: Zielfunktion für unterschiedliche y-Werte

In Abbildung 4.18 ist die Zielfunktion $w_{z\,max} \cdot A$ für y-Werte von $y = 0\,mm$ bis $y = 20\,mm$ dargestellt. Es ist also der ganze zulässige Bereich von y diskret abgebildet inklusive y = 0 mm und seinen Grenzen $1\,mm < y < b = 20\,mm$. Der zulässige Bereich für z ist strichliert eingezeichnet. Alle Graphen gehen durch den Punkt $z = 0\,mm$, $w_{z\,max} \cdot A = 595{,}2$ mm^3 und den Punkt $z = h/2 = 15\,mm$, $w_{z\,max} \cdot A = 595{,}2\,mm^3$. Je kleiner y wird, desto näher rückt das Minimum Richtung $z = 0\,mm$ und der Funktionswert wird immer kleiner. Das Optimum liegt also beim kleinstmöglichen y-Wert, also bei $y = 1\,mm$. Das heißt, die Lage des Optimums deckt sich mit jener der „Optimierungsaufgabe 1“, wo fix vorgegeben ist, dass y den Wert $1\,mm$ hat und das Optimum bei $z_{opti} = 2{,}26\,mm$ liegt. Das gleiche Verhalten ist auch im 3D Diagramm der Abbildung 4.19 zu erkennen.

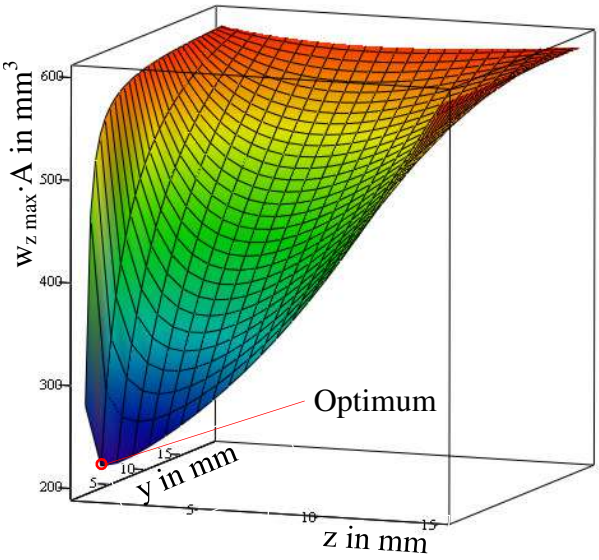


Abbildung 4.19: 3D Darstellung von der Zielfunktion für die „Optimierungsaufgabe 5“

Bei der Optimierung mit einer Variable waren alle Optimierer erfolgreich, siehe „Optimierungsaufgabe 1“, bei der Optimierung mit zwei Variablen trifft das nicht zu. Sehr gut und schnell hat der lokale Optimierer *coliny_cobyla* funktioniert. Wie in Tabelle 4.10 ersichtlich, ist der relative Fehler für die Variablen z und y und auch für die Zielfunktion hier kleiner als 1 %. Die globalen Optimierer kommen nur auf Ergebnisse, welche in der Nähe der analytischen Lösung liegen. Grün markiert sind jene Felder in Tabelle 4.10, bei denen der relative Fehler kleiner als 1 % ist. Für die Berechnung der relativen Fehler der Variablen wird das analytische Optimum von $y = 1\,mm$ und $z = 2{,}26\,mm$ verwendet. Bei der Zielfunktion wird für die Berechnung des relativen Fehlers das Ergebnis von Abaqus, bei dem für z und y die analytisch ermittelten optimalen Werte eingesetzt werden, verwendet.

Tabelle 4.10: Ergebnis der „Optimierungsaufgabe 5“

Optimierer von Dakota	z_{opti} in mm	y_{opti} in mm	$w_{z\,max} \cdot A$ in mm^3	relativer Fehler in %			Zeit
				z_{opti}	y_{opti}	$w_{z\,max} \cdot A$	
coliny_cobyla	2,26	1,00	277,15	0,05	0,00	0,00	11 min
coliny_cobyla	2,26	1,00	277,15	0,05	0,00	0,00	11 min
efficient_global	2,33	1,00	277,19	3,14	0,00	0,01	8 min
efficient_global	2,33	1,00	277,19	3,14	0,00	0,01	6 min
soga	2,36	1,05	279,21	4,28	5,07	0,74	5 h 48 min
soga	2,32	1,07	279,78	2,66	6,55	0,95	6 h 3 min
coliny_direct	3,22	1,00	282,16	29,86	0,01	1,81	1 h 50 min
coliny_direct	3,22	1,00	282,16	29,86	0,01	1,81	1 h 20 min
coliny_ea	1,81	1,08	282,71	25,00	8,06	2,01	1 h 54 min
coliny_ea	3,63	1,06	288,17	37,74	6,29	3,97	1 h 57 min

Optimierungsaufgabe 6, w_z limitiert

Der optimale Wert z_{opti} für die Gurtdicke z und y_{opti} für die Stegdicke y sollen ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinstmöglich sind. Zusätzlich ist auch noch ein konkreter Wert von 2 mm bzw. 1,97 mm für die Durchbiegung gegeben, welcher nicht überschritten werden darf. Wie im 3D Diagramm der Zielfunktion in Abbildung 4.20 ersichtlich, treten die kleinsten Werte für die Zielfunktion bei kleinen y-Werten auf. Daher werden die Zielfunktion und die Durchbiegung für kleine y-Werte auch noch zusätzlich im 2D Diagramm dargestellt. Die Gitterebene im 3D Diagramm der Durchbiegung stellt das Limit $w_z \leq 2\,mm$ dar. Der zulässige Bereich liegt unterhalb dieser Ebene. Im 2D Diagramm, Abbildung 4.21, wird diese Grenze durch horizontale Geraden dargestellt. Die vertikalen Linien stellen die Begrenzungen dar, welche sich dadurch für die Zielfunktion bei unterschiedlichen y-Werte ergeben. In diesem Diagramm kann man auch erkennen, dass der minimale Wert ($y = 1\,mm / z = 2{,}26\,mm$) der Zielfunktion, außerhalb des zulässigen Bereichs liegt und das Optimum bei $y = 1\,mm$ und bei $w_z = 2\,mm$ liegt, also genau an der Grenze.

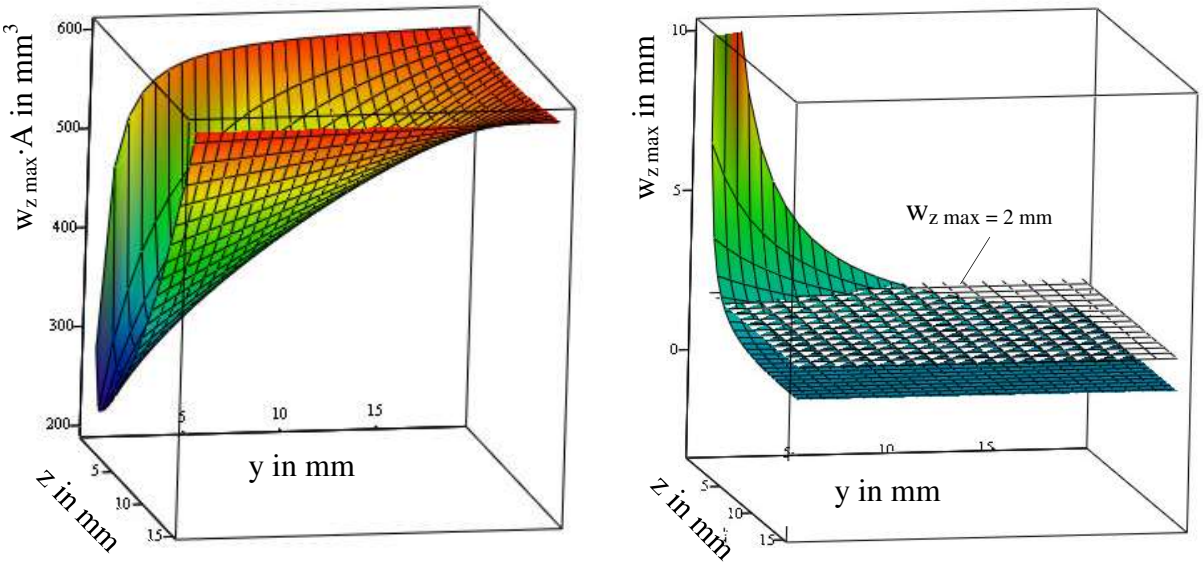


Abbildung 4.20: 3D Darstellungen von der Zielfunktion und der Durchbiegung für die „Optimierungsaufgabe 6“, mit der Restriktion $w_z \leq 2\,mm$

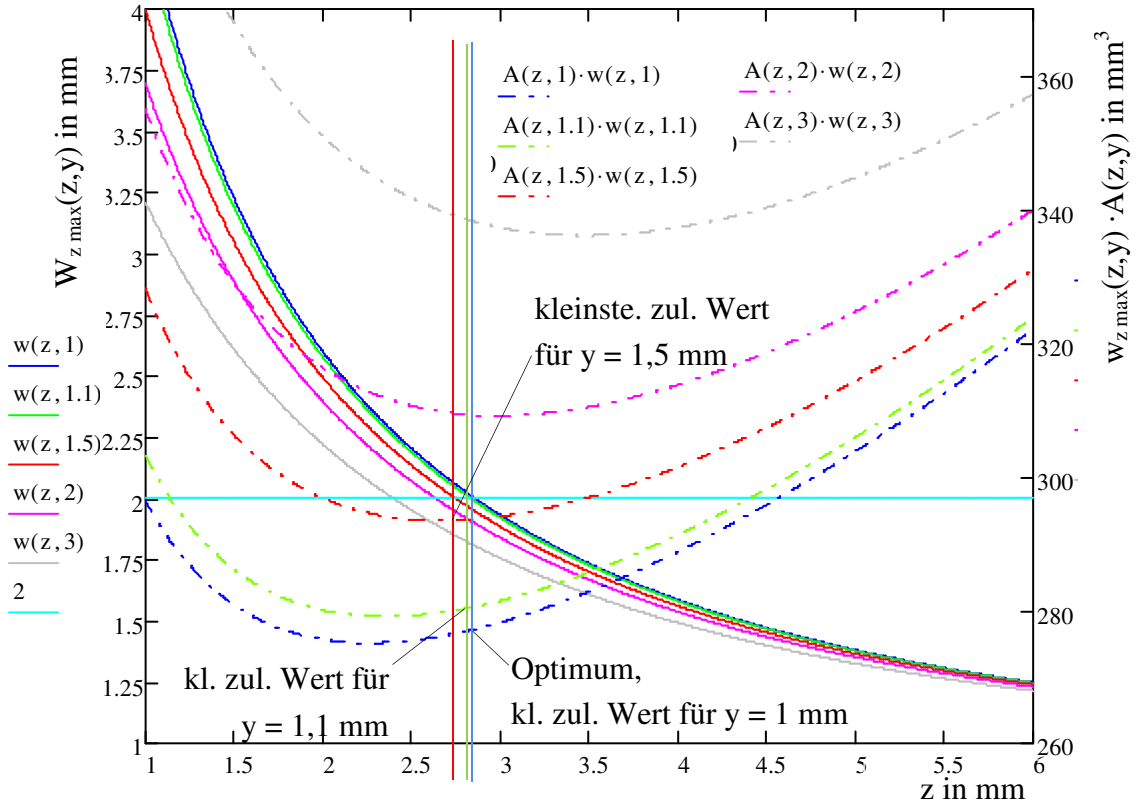


Abbildung 4.21 Graphische Darstellung des Ergebnis von der „Optimierungsaufgabe 6“

Durchgeführt wird diese Optimierungen wieder nur mit den beiden Optimieralgorithmen *coliny_cobyla* und *efficient_global*. Für die Grenze von $w_z \leq 2\,mm$ liefern beide ein gutes Ergebnis. Bei dem Limit von 1,97 mm kommt *efficient_global* nicht genau auf das Optimum, wie in Tabelle 4.11 ersichtlich ist.

Tabelle 4.11: Ergebnis der „Optimierungsaufgabe 6“

Optimierer von Dakota	z_{opti} in mm	y_{opti} in mm	$w_{z\,max}\cdot A$ in mm^3	$w_{z\,max}$ in mm		Zeit in min
				Ergebnis	Limit	
coliny_cobyla	2,89	1,00	279,48	2,00	≤ 2	7
efficient_global	2,89	1,00	279,48	2,00	≤ 2	6
coliny_cobyla	2,95	1,00	279,91	1,97	≤ 1.97	6
efficient_global	2,96	1,00	280,01	1,96	≤ 1.97	6

Optimierungsaufgabe 7, σ_x limitiert

Der optimale Wert z_{opti} für die Gurtdicke z und y_{opti} für die Stegdicke y sollen ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinst möglich sind. Zusätzlich ist auch noch ein konkreter Wert von 400 N/mm^2 für die Spannung gegeben, welcher nicht überschritten werden darf. Grafisch schaut dieses Problem ähnlich aus, wie das vorherige. Im 3D Diagramm der Spannung in Abbildung 4.22 ist wieder das Limit $\sigma_x \leq 400\,N/mm^2$ mit einer Gitterebene eingezeichnet. Im 2D Diagramm, Abbildung 4.23,

sind die Zielfunktion und die Spannung für kleine y-Werte abgebildet. Hier wird die Restriktion durch eine horizontale Linie dargestellt. Die vertikalen Linien stellen die Begrenzungen dar, welche sich dadurch für die Zielfunktion bei unterschiedliche y-Werte ergeben. Dabei ist gut ersichtlich, dass das Optimum beim kleinstmöglichen y-Wert und genau an der Spannungsgrenze liegt.

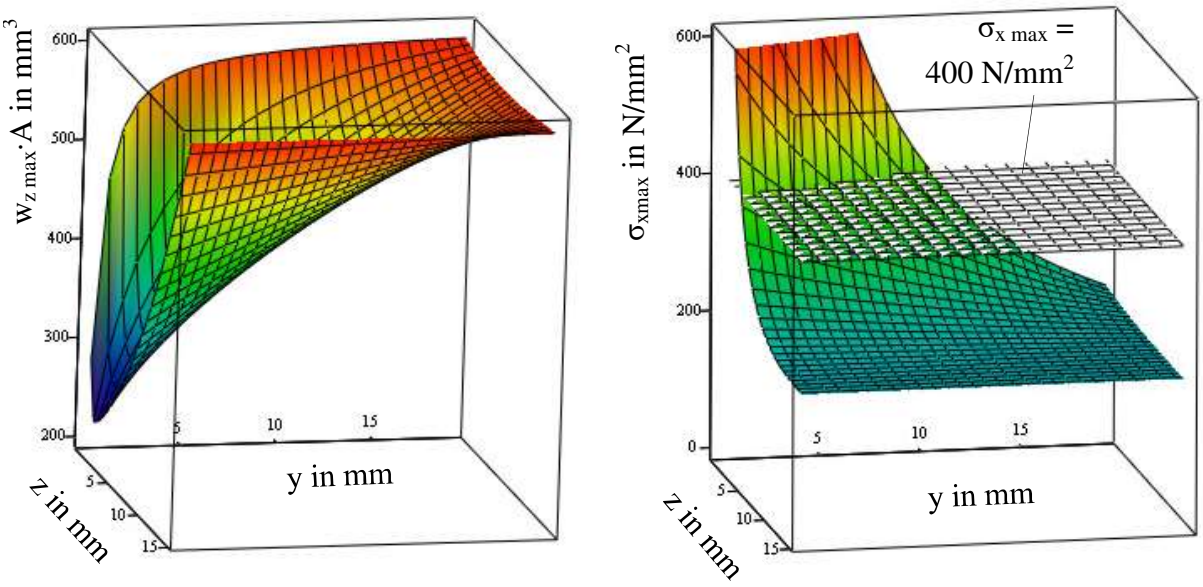


Abbildung 4.22: 3D Darstellungen der Zielfunktion und der Spannung für die „Optimierungsaufgabe 7“, mit der Restriktion $\sigma_x \leq 400\ N/mm^2$

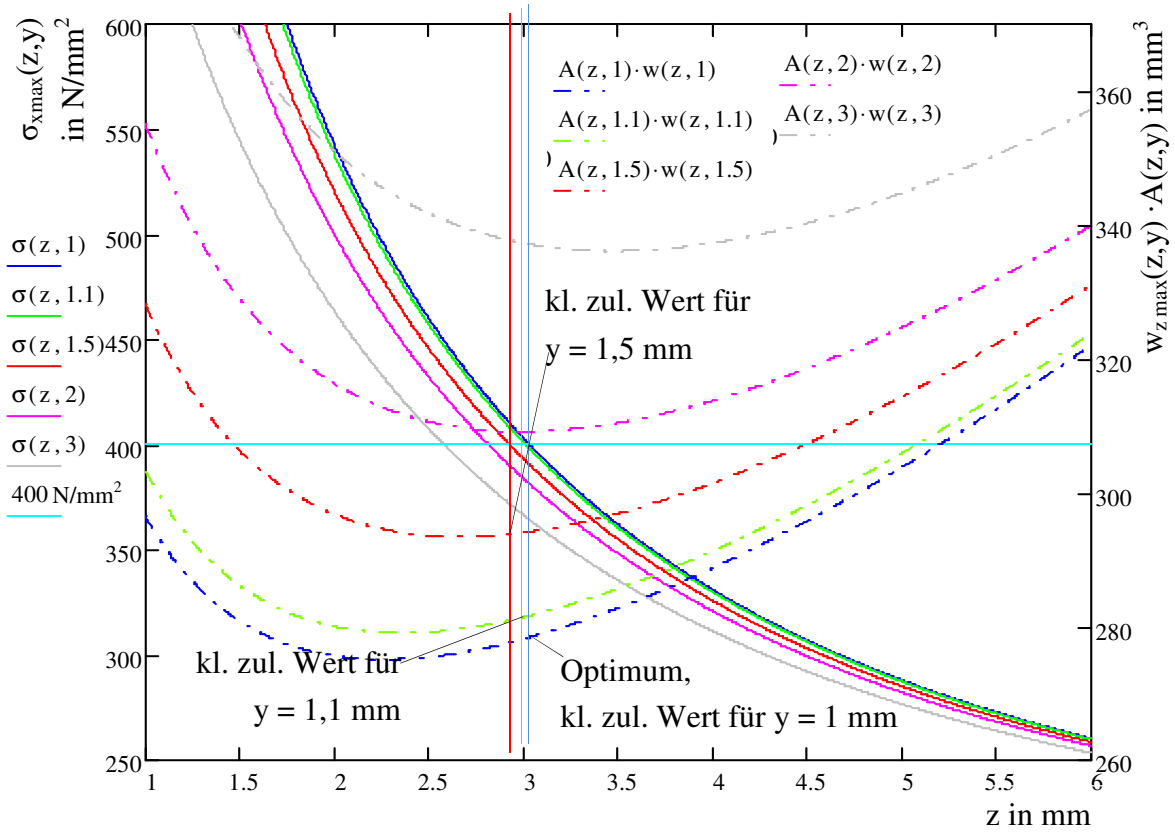


Abbildung 4.23: Graphische Darstellung des Ergebnisses der „Optimierungsaufgabe 7“

In der Tabelle 4.12 sind die Ergebnisse zusammengefasst. Der lokale Optimierer *coliny_cobyla* kommt auf das richtige Ergebnis, während der lokale Optimierer *efficient_global* nur ein Ergebnis in der Nähe des Optimums angibt.

Tabelle 4.12. Ergebnis der „Optimierungsaufgabe 7“

Optimierer von Dakota	z_{opti} in mm	y_{opti} in mm	$w_{z\,max}\cdot A$ in mm^3	$\sigma_{x\,max}$ in N/mm^2	
				Ergebnis	Limit
coliny_cobyla	3,03	1,00	280,54	400,00	≤ 400
coliny_cobyla	2,57	1,00	277,79	450,00	≤ 450
efficient_global	2,58	1,00	277,80	449,77	≤ 450
efficient_global	3,06	1,00	280,72	397,83	≤ 400

Optimierungsaufgabe 8, σ_x und w_z limitiert

Der optimale Wert z_{opti} für die Gurtdicke z und y_{opti} für die Stegdicke y sollen ermittelt werden, so dass die Spannungen, die Durchbiegungen und die Masse kleinstmöglich sind. Zusätzlich sind für die Spannung und die Durchbiegung auch noch konkrete Werte gegeben, welche nicht überschritten werden dürfen. Die „Optimierungsaufgabe 8“ ist eine Kombination aus den beiden vorherigen Aufgaben. Im 2D Diagramm in der Abbildung 4.24 sind die Spannung, die Durchbiegung und die Zielfunktion für kleine y-Werte dargestellt. Bei der „Optimierungsaufgabe 6“, bei der nur die Durchbiegung limitiert ist, wird bei einer Grenze von 2 mm das Optimum bei z_{opti} = 2,89 mm und y_{opti} = 1 mm gefunden. Wenn man diese

Ergebnisse mit jenen von der „Optimierungsaufgabe 7“ vergleicht, bei der das Optimum bei einer Begrenzung der Spannung auf 450 N/mm^2 bei $z_{opti} = 2{,}57\,mm$ und $y_{opti} = 1\,mm$ liegt, sieht man, dass dieses Optimum außerhalb des zulässigen Bereichs von „Optimierungsaufgabe 6“ liegt. Das Optimum liegt also bei der „Optimierungsaufgabe 8“ an der gleichen Stelle wie bei der „Optimierungsaufgabe 6“. In Abbildung 4.24 ist diese Tatsache gut ersichtlich, die vertikalen Linien kennzeichnen die Restriktionen für $y = 1\,mm$.

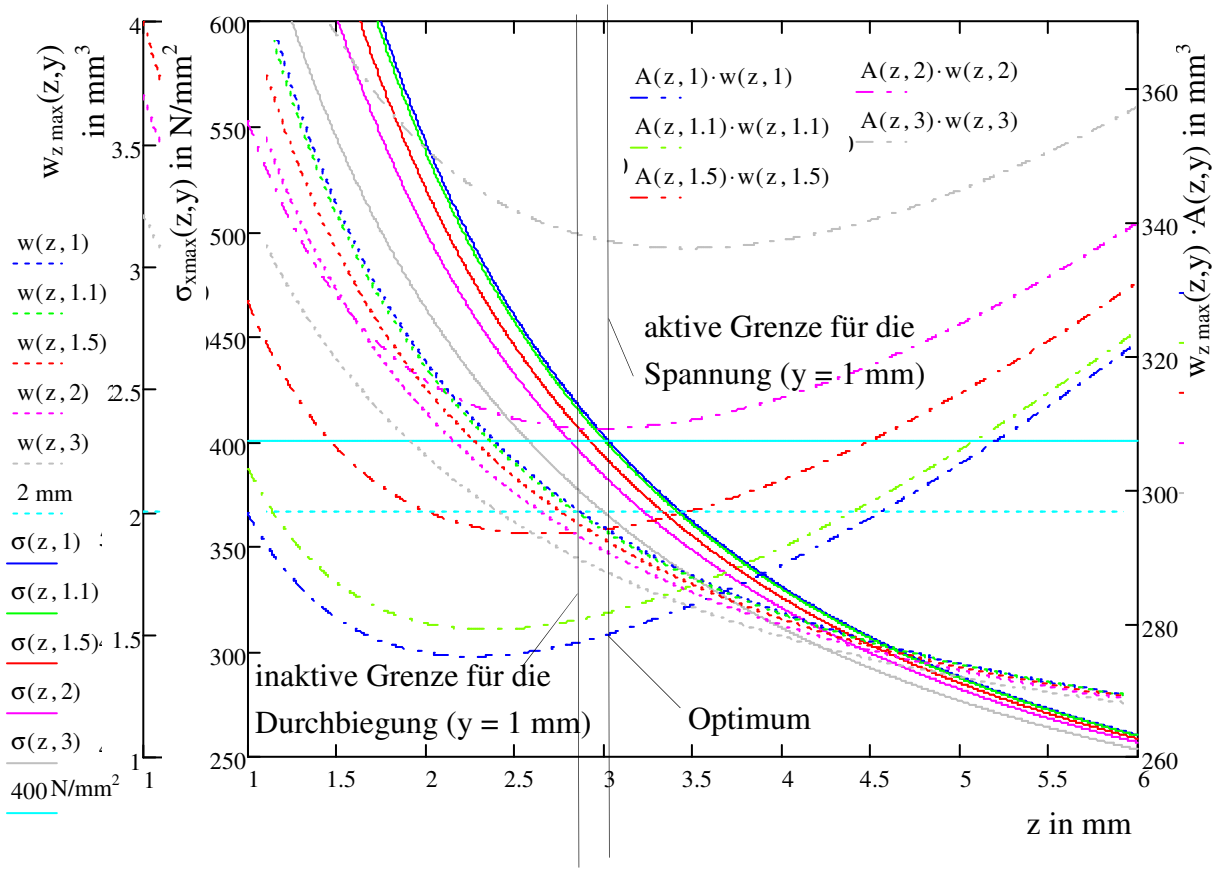


Abbildung 4.24: Graphische Darstellung des Ergebnisses der „Optimierungsaufgabe 8“

In der Tabelle 4.13 sind die Ergebnisse zusammengefasst. Wie auch schon bei den beiden vorherigen Aufgaben hat nur der Optimierer *coliny_cobyla* das Optimum gefunden.

Tabelle 4.13: Ergebnis der „Optimierungsaufgabe 8“

Optimierer von Dakota	z_{opti} in mm	y_{opti} in mm	$w_{z\,max} \cdot A$ in mm^3	$w_{z\,max}$ in mm	$\sigma_{x\,max}$ in N/mm^2	Limit in mm bzw N/mm^2
efficient_global	2,92	1,00	279,72	1,98	410,54	$w_z \leq 2$ $\sigma_x \leq 450$
coliny_cobyla	2,89	1,00	279,48	2,00	414,13	$w_z \leq 2$ $\sigma_x \leq 450$

4.2 Sandwich-Übergang

Sandwichbauteile bestehen meist aus zwei Deckschichten und einem Kern, wie in Kapitel 2.3 beschrieben. Da bei der Formoptimierung des I-Trägers das Ergebnis von Dakota und Abaqus mit dem analytischen Ergebnis übereinstimmt, also erfolgreich war, wird als zweites Beispiel ein Sandwichelement, bei welchen die Deckschichten zusammengeführt werden, optimiert. Wie in Bild a) von Abbildung 2.8 und in Bild b) von Abbildung 2.7 dargestellt, kann diese Zusammenführung aufgrund einer eingeleiteten Last, oder aufgrund von einer notwendigen Kantenversteifung am Ende eines Sandwichbauteiles erfolgen.

Die Geometrie des Übergangs ist in Abbildung 4.25 dargestellt, genauso wie seine Aufteilung in drei Bereiche. Die beiden äußeren Deckschichten sind aus Aluminium und der Kern ist aus dem Schaum Airex R82.60. Die einzigen fix vorgegebenen Abmessungen sind die Breite b des Bauteiles mit 10 mm, die Höhe h des Kerns mit 10 mm und die gesamte Länge l_{ges} mit 75 mm. Mögliche Variablen sind der Winkel α des Kerns und die beiden Dicken der Deckschichten t_1 und t_2. Es gibt zwei Möglichkeiten l_1 und l_3 zu definieren. Bei „Modell 1“ ist die Länge l_1 mit 12 mm fix vorgegeben und die Länge l_3 ergibt sich über den Winkel α. Bei „Modell 2“ ist es genau umgekehrt l_3 ist mit 12 mm definiert und die l_1 ergebt sich über den Winkel α. Am breiten Ende ist der Übergang eingespannt und am dünnen Ende greift ein Moment mit 500 Nmm an.

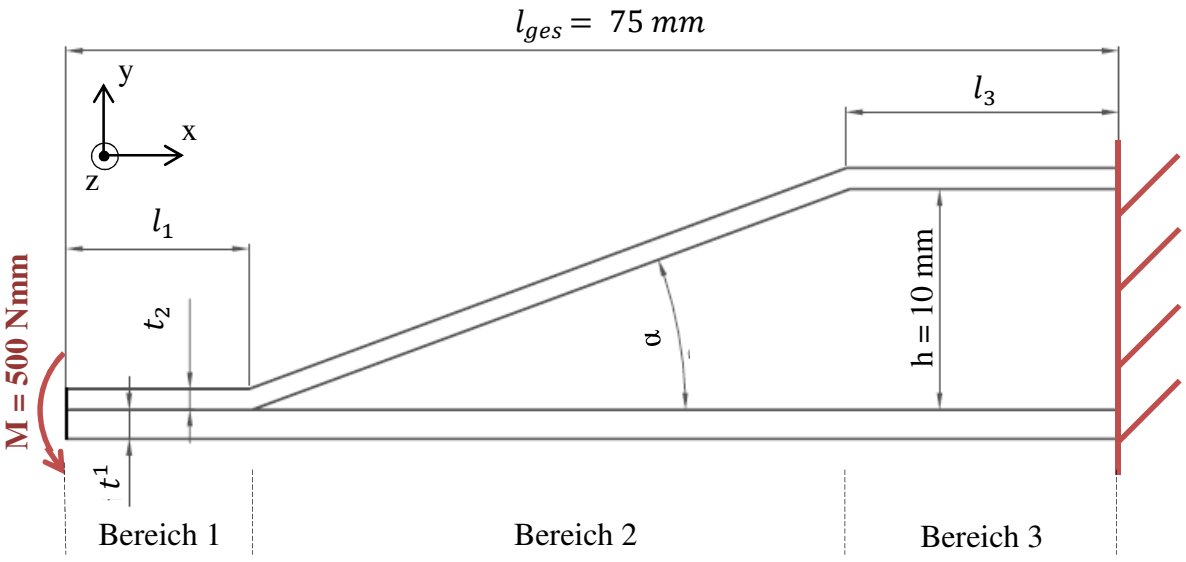


Abbildung 4.25: Abmessungen des Übergangs

Der zulässige Bereich der möglichen Variablen ist wie folgt festgelegt:

$$10° < \alpha < 20°$$
$$0{,}5\ mm < t1 < 3\ mm \qquad (35)$$
$$0{,}5\ mm < t2 < 3\ mm$$

Vier unterschiedliche Fälle sollen optimiert werden. Bei „Fall 1“ soll mit der Variable α die Durchbiegung minimiert werden bei einer Obergrenze von 110 N/mm^2 für die Mises

Spannung. Bei „Fall 2“ wird zusätzlich zur Spannung auch noch Masse auf 4.3 g limitiert. Bei „Fall 3“ bzw. „Fall 4“ sind die Aufgabenstellungen gleich wie bei „Fall 1“ bzw. „Fall 2“, nur statt einer Variable werden die drei Variablen α, t_1 und t_2 verwendet.

4.2.1 Finite-Elemente-Modell

Wie auch schon beim I-Träger ist das Finite-Elemente-Netz so aufgebaut, dass es bei Änderungen der Abmessungen mitskaliert wird. Der Aufbau der Textdatei, in welcher die Modellgeometrie, die Netzgeometrie und die Randbedingungen festgelegt sind, entspricht jenem des I-Trägers. Da es sich um ein parametergesteuertes Finite-Elemente-Modell handelt, sind in dieser Datei nur die Werte für die Konstanten definiert. Die Variablen werden aus der Datei *ParInput.inp* eingelesen.

Der Übergang ist mit 3D Elementen aufgebaut, welche nur translatorische Freiheitsgrade besitzen. Um am linken Rand in Abbildung 4.26, ein Moment einleiten zu können, werden am Ende des dünnen Bereichs Schalenelemente angebracht, an welchen ein Moment angreifen kann.

Da das Koordinatensystem aufgrund des Winkels nicht immer mit der Richtung, in welcher die maximalen Spannungen auftreten, übereinstimmt, wurde die Mises Spannung σ_v als Bezugswert verwendet. Bei der Verschiebung wird jene in die y-Richtung betrachtet.

Analog zum I-Träger werden hauptsächlich C3D8R Elemente verwendet. Für die Modellierung des Kernes kommen zusätzlich C3D6 Elemente in Keilform zur Anwendung und bei der Schale handelt es sich um S4R Elementen.

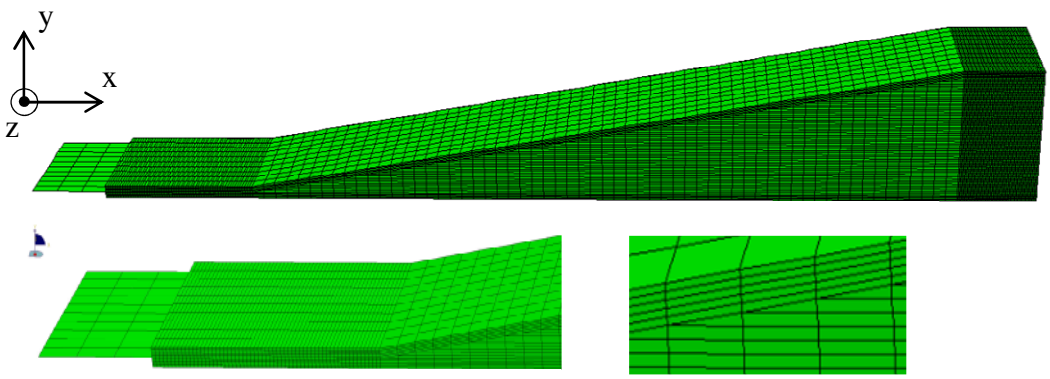


Abbildung 4.26: Finite-Elemente-Modell vom Übergang

Material

Im 3D Modell werden für den Werkstoff Aluminium, wie auch für den Schaum Airex R82.60, ein elastisches isotropes Material verwendet. Der thermoplastische Hartschaumstoff Schaum Airex R82.60 ist geeignet für strukturelle Leichtbauanwendungen mit hohen Brandschutzanforderungen. Er wird zum Beispiel für Rumpf- und Flügelteile bei Sportflugzeugen oder bei der Inneneinrichtung von Schienen- und Straßenfahrzeugen verwendet[3]. In Abbildung 4.27 sind die Materialkennwerte des verwendeten Schaums Airex

R82.60 angegeben. Jene Werte, welche für die Optimierung benötigt werden sind rot eingekreist.

AIREX

CORE MATERIALS

MECHANISCHE EIGENSCHAFTEN						
Typische Daten für AIREX® R82		Einheit (metrisch)	Wert[1)]	R82.60	R82.80	R82.110
Dichte	ISO 845	kg/m³	Mittelwert *Typ. Bereich*	60 *54 - 69*	80 *72 - 95*	110 *99 - 126*
Druckfestigkeit senkrecht	ISO 844	N/mm²	Mittelwert *Minimum*	0.70 *0.60*	1.1 *0.9*	1.4 *1.2*
Druckmodul senkrecht	DIN 53421	N/mm²	Mittelwert *Minimum*	46 *40*	62 *56*	83 *60*
Zugfestigkeit in Plattenebene	ISO 527 1-2	N/mm²	Mittelwert *Minimum*	1.7 *1.2*	2.0 *1.7*	2.2 *1.9*
Zugmodul in Plattenebene	ISO 527 1-2	N/mm²	Mittelwert *Minimum*	45 *35*	54 *50*	64 *54*
Schubfestigkeit	ISO 1922	N/mm²	Mittelwert *Minimum*	0.80 *0.65*	1.1 *0.9*	1.4 *1.15*
Schubmodul	ASTM C393	N/mm²	Mittelwert *Minimum*	18 *15*	23 *20*	30 *25*
Schubbruchdehnung	ISO 1922	%	Mittelwert *Minimum*	25 *15*	23 *15*	18 *10*
Schlagzähigkeit	DIN 53453	kJ/m²	Mittelwert	1.0	1.3	1.4
Wärmeleitfähigkeit bei Raumtemperatur	ISO 8301	W/m.K	Mittelwert	0.036	0.037	0.040
Standardplatte	Breite	mm ± 5		1350	1200	1000
	Länge	mm ± 5		2800	2700	2300
	Dicke	mm ± 0.5		3 bis 60	3 bis 60	5 bis 30
Farbe				gebrochen weiss	gebrochen weiss	gebrochen weiss

Abbildung 4.27: Materialkennwerte des Schaumes Airex R82.60 [3]

Für Aluminium werden der E-Module mit 70000 N/mm^2 angegeben und die Poissonzahl mit 0,33. Für den Schaum wurde der Zugmodul in Plattenebene mit 35 N/mm^2 als E-Modul verwendet, da die Biegespannungen in Plattenebenen auftreten. Mit diesem Zugmodul und dem Schubmodul G von 15 N/mm^2 wird mit der Formel $G = \frac{E}{2(1+\nu)}$ eine Poissonzahl von 0,167 ermittelt. Für die Berechnung der Masse wird die Dichte des Schaumes mit 60 kg/m^3 verwendet und für das Aluminium eine Dichte von 2700 kg/m^3 [3].

Unterschied drei oder fünf Elemente je Deckschicht

Zuerst wird ein Modell mit drei Elementen über die Dicke der Deckschicht erstellt. Zur Erzielung eines genaueren Ergebnisses, wird die Anzahl der Elemente über die Deckschichtdicke bei einem zweiten Modell von drei auf fünf erhöht. Um überprüfen zu können, ob es zu einer Verbesserung kommt, werden die maximalen Werte der Mises Vergleichsspannung $\sigma_{v\ max}$ und der Durchbiegung in y-Richtung $w_{y\ max}$ für unterschiedliche α-Werte verglichen, siehe Tabelle 4.14. Die Deckschichtdicke für „Modell 1“ und „Modell 2“ beträgt 1 mm.

Tabelle 4.14: Unterschiede aufgrund der Elementanzahl in der Deckschicht

Modell 1, l_1 = konstant								
	5 Elemente		3 Elemente		Unterschied			
					absoluter		relativer in %	
α in °	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²
10	0,58	98,65	0,60	89,10	0,02	9,55	3	10
12	0,56	102,27	0,58	91,70	0,02	10,56	3	10
14	0,55	105,47	0,57	94,61	0,02	10,86	3	10
16	0,54	108,37	0,56	97,79	0,02	10,58	4	10
18	0,54	111,03	0,56	100,62	0,02	10,40	4	9
20	0,53	113,51	0,55	103,21	0,02	10,31	4	9
Modell 2, l_3 = konstant								
	5 Elemente		3 Elemente		Unterschied			
					absoluter		relativer in %	
α in °	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²
10	0,34	99,19	0,35	88,80	0,01	10,38	3	10
12	0,75	102,32	0,77	91,77	0,02	10,55	3	10
14	1,09	105,58	1,12	94,58	0,03	11,00	3	10
16	1,37	108,52	1,41	97,70	0,04	10,82	3	10
18	1,60	111,22	1,66	100,46	0,05	10,76	3	10
20	1,80	113,76	1,86	102,97	0,06	10,79	3	9

Der Vergleich der Durchbiegungen w_y zeigt, dass die Modelle mit den fünf Elementen je Deckschicht steifer sind als jene mit drei Elementen. Der relative Unterschied, bezogen auf die Durchbiegung mit fünf Elementen je Deckschicht, beträgt zwischen 3 % und 4 %. Die Werte für die maximalen Mises Spannungen werden mit steigender Elementanzahl größer, da die Spannungserhöhung im Übergang zwischen Bereich 1 und 3 besser dargestellt werden kann. Dieses Verhalten ist nicht nur in der Tabelle ersichtlich, sondern auch in den Bildern von Abbildung 4.28 und in den Diagrammen in den Abbildung 4.29 und Abbildung 4.30. Für alle weiteren Berechnungen wurde das Modell mit den 5 Elementen verwendet.

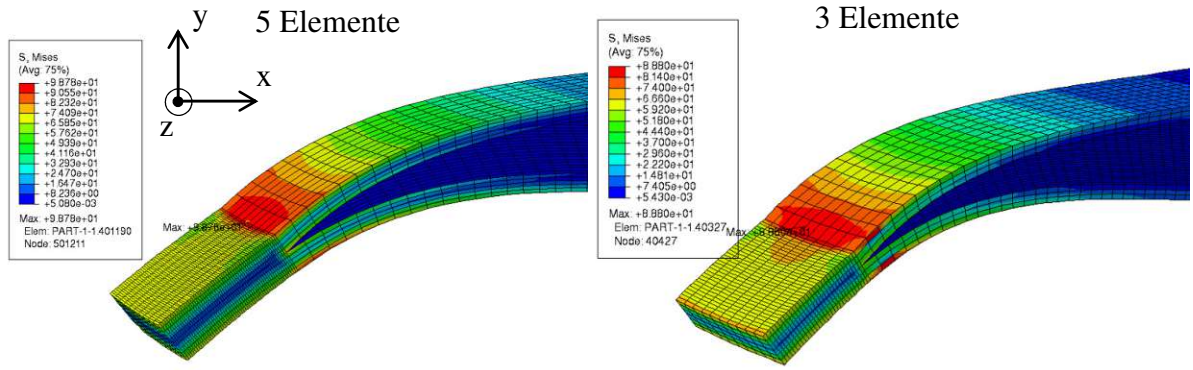


Abbildung 4.28: Darstellung der Spannungserhöhung zwischen „Bereich 1“ und „Bereich 3“ für α = 10°

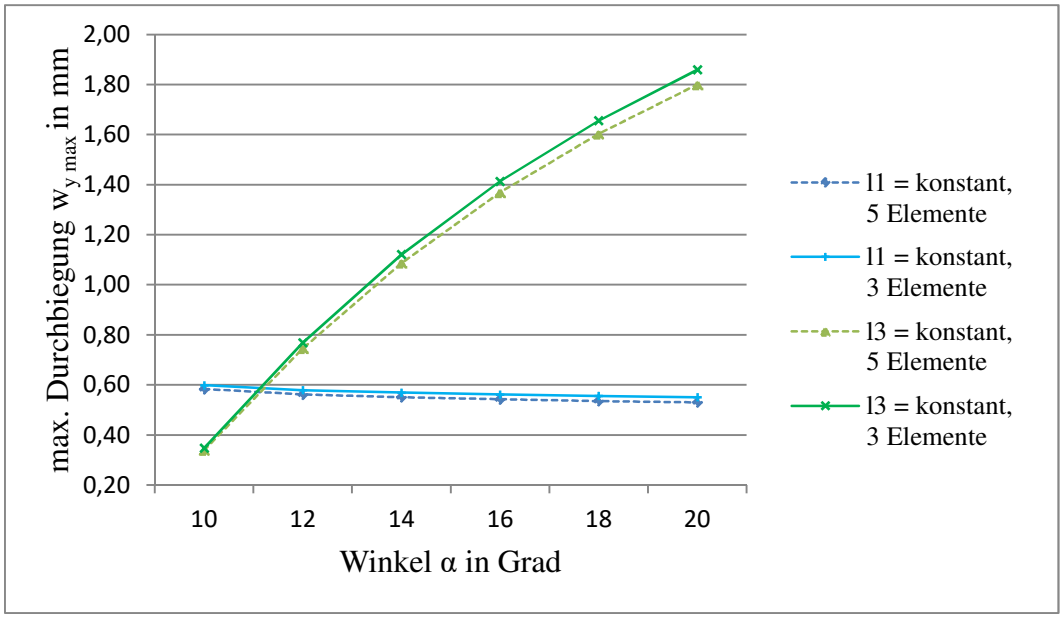


Abbildung 4.29: Vergleich der Durchbiegungen mit 3 und 5 Elementen über die Dicke der Deckschicht

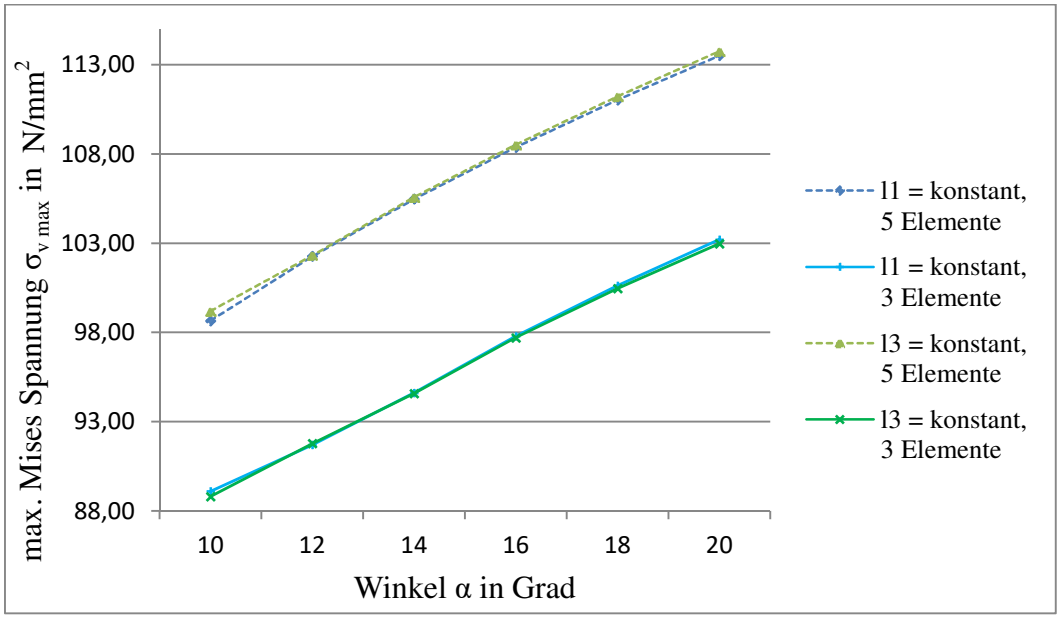


Abbildung 4.30: Vergleich der Spannungen mit 3 und 5 Elementen über die Dicke der Deckschicht

Überprüfung des Modells

Um zu überprüfen, ob das Finite-Elemente-Model richtig aufgebaut ist und ein Verhalten zeigt, welches erwartet wird, wird überprüft, ob sich das Modell im ungestörten Bereich des dünnen Bereichs wie ein Biegeträger mit konstantem Querschnitt verhält. Als Referenzmodell wird das Modell 1 mit den Abmessungen $t_1 = t_2 = 1\,mm$ und $\alpha = 20°$ verwendet.

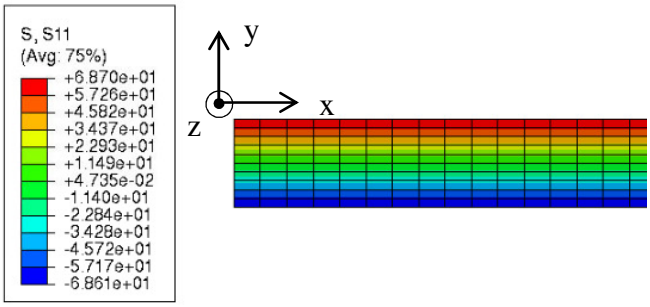


Abbildung 4.31: Normalspannung $\sigma_x = S_{11}$ im ungestörten Bereich

In Abbildung 4.31 ist die Normalspannung in x-Richtung σ_x im ungestörten Bereich von „Bereich 1“ des „Modells 1“ dargestellt. Es ist gut ersichtlich, dass sich im ungestörten Bereich der zu erwartende lineare Spannungsverlauf einstellt. Die maximale Normalspannung $\sigma_{x\,max}$ tritt in den äußersten Elementen, also der Randfaser, auf. Zur Kontrolle wird die maximale Spannung von „Bereich 1“ analytisch für den Integrationspunkt des äußersten Elements berechnet. Da der Integrationspunkt bei den verwendeten C3D8R Elementen in der Mitte liegt, ergibt sich ein Randfaserabstand e von 0,9 mm. Der analytische Wert für die maximale Spannung beträgt 67,5 N/mm^2.

$$\sigma_{x\,max} = \frac{M \cdot e}{I} = \frac{M \cdot 12 \cdot e}{b \cdot h^3} = \frac{500\ Nmm * 0{,}9\ mm * 12}{10\ mm * (2\ mm)^3} = 67{,}5\ \frac{N}{mm} \qquad (36)$$

Tabelle 4.15: Vergleich der analytischen und numerischen Lösung für $\sigma_{x\,max}$ von „Bereich 1“

	analy. Lösung	l_3 = konst, $\alpha = 20°$		l_3 = konst, $\alpha = 10°$		l_1 = konst, $\alpha = 10°$		l_1 = konst, $\alpha = 20°$	
Element	$\sigma_{x\,max}$ in N/mm^2	$\sigma_{x\,max}$ in N/mm^2	rel. Fehler in %	$\sigma_{x\,max}$ in N/mm^2	rel. Fehler in %	$\sigma_{x\,max}$ in N/mm^2	rel. Fehler in %	$\sigma_{x\,max}$ in N/mm^2	rel. Fehler in %
610 (oberes)	67,5	67,8	0,44	67,75	0,37	66,3	1,78	68,18	1,01
601 (unteres)	-67,5	-68,43	1,38	-68,58	1,60	-68,33	1,23	-68,18	1,01

In Tabelle 4.15 ist ersichtlich, dass der Unterschied zwischen der analytischen Lösung und der numerischen kleiner als 2 % ist.

Um zu überprüfen, ob es zu keinem Klaffen zwischen den Elementen kommt, wird eine Berechnung durchgeführt, bei welcher der Kern und die Deckschicht aus Aluminium sind.

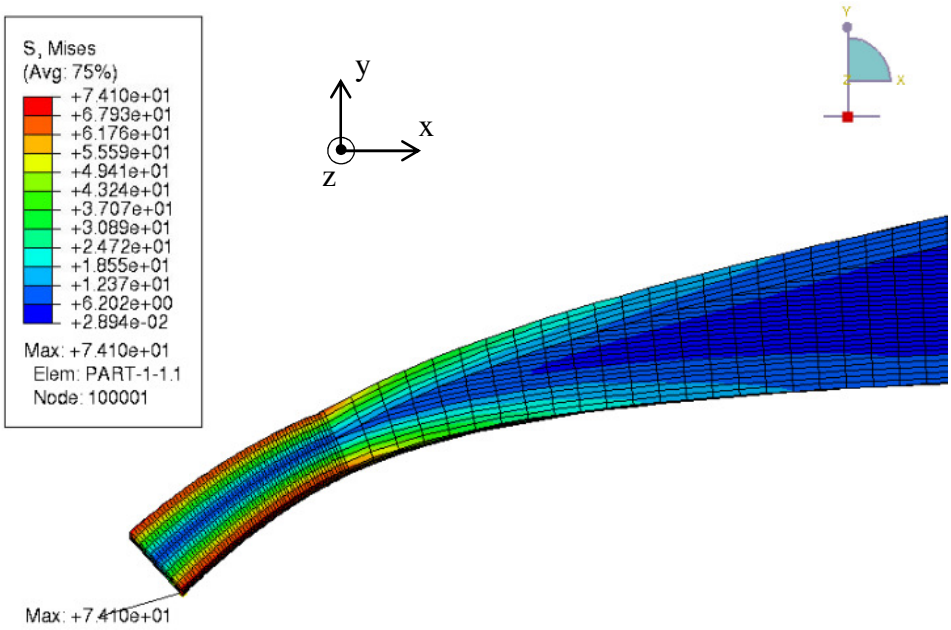


Abbildung 4.32: nur Aluminium, l_3 = konstant, α = 10°, $t_1 = t_2 = 1\,mm$

Wie in Abbildung 4.32 ersichtlich, stellt sich der erwartete Spannungsverlauf ein. Die maximalen Spannungen treten am Rand auf und, wenn der Querschnitt höher und dadurch auch das Flächenträgheitsmoment größer wird, werden die auftretenden Spannungen kleiner.

Unterschied l_1= konstant oder l_3 = konstant

Um das unterschiedliche Verhalten von „Modell 1“ mit konstanten l_1 und von „Modell 2“ mit konstanten l_3 besser zu erkennen, wurde eine Parameterstudie für unterschiedliche α-Werte durchgeführt. Die Ergebnisse der unterschiedlichen Modelle findet man in Tabelle 4.16 und die grafische Darstellung in Abbildung 4.36. Es wird jenes Modell mit fünf Elementen über die Schichtdicke verwendet, und eine Deckschichtdicke von 1 mm. Verglichen wird die Masse m des gesamten Übergangs, die maximal auftretende Mises Spannung $\sigma_{v\,max}$und die maximale Durchbiegung in y-Richtung $w_{y\,max}$.

Tabelle 4.16: Gegenüberstellung von „Modell 1“ und „Modell 2“

	l_1 = konstant			l_3 = konstant		
α in °	m in g	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	m in g	$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²
10	4,26	0,583	98,65	4,29	0,339	99,19
12	4,29	0,562	102,27	4,26	0,745	102,32
14	4,31	0,551	105,47	4,24	1,087	105,58
16	4,32	0,542	108,37	4,23	1,369	108,52
18	4,34	0,535	111,03	4,21	1,603	111,22
20	4,35	0,530	113,51	4,20	1,800	113,76

Wie in Abbildung 4.33 ersichtlich, unterscheiden sich die maximalen Mises Spannungen der beiden Modelle nur minimal, da sie in der Ecke im Übergang von „Bereich 1“ zu „Bereich 2“ auftreten und daher vor allem vom Winkel abhängen. Bei gleichem Winkel ist die Form

dieser Ecke bei beiden Modellen gleich, unterschiedlich ist nur die Lage der Ecke in x-Richtung. Mit steigendem Winkel wird diese Ecke immer spitzer und daher nimmt auch die maximale Spannung zu.

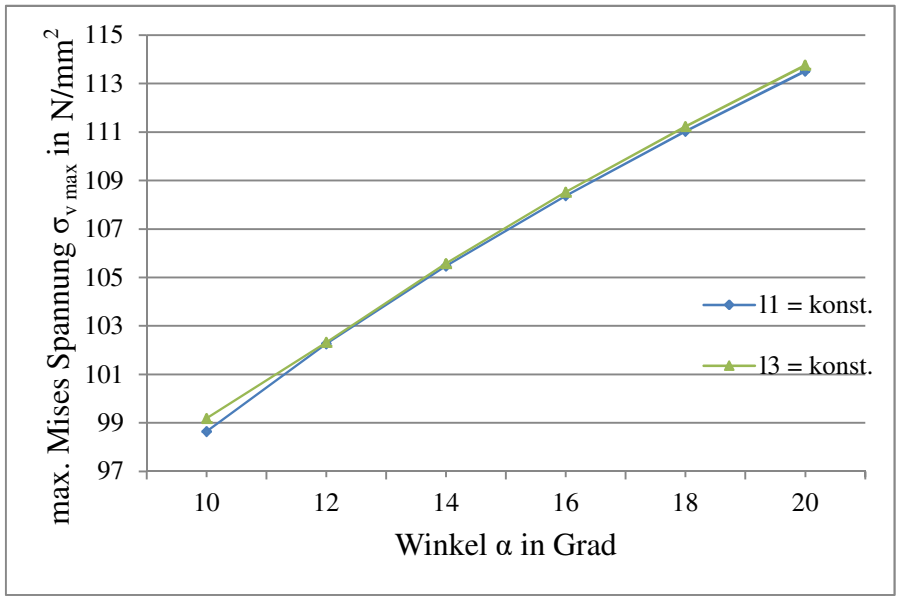


Abbildung 4.33: Vergleich der maximalen Mises Spannung von „Modell 1“ und „Modell 2“

In Bezug auf die Masse verhalten sich die beiden Modelle gegenläufig, siehe Abbildung 4.34. Bei konstantem l_1 wird mit steiler werdendem Winkel der „Bereich 3“ größer. Da ein Stück von „Bereich 3“ schwerer ist als ein gleich breites Stück vom „Bereich 2“, wird das Bauteil mit steigendem Winkel schwerer. Wenn l_3 konstant ist, ist es genau umgekehrt, da hier ein Stück von „Bereich „ leichter ist als ein gleich langes Stück von „Bereich 2“.

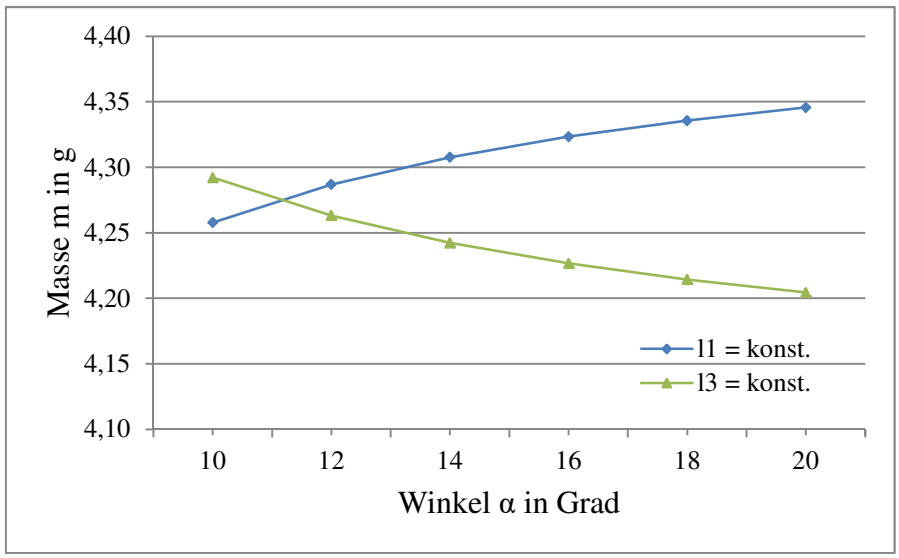


Abbildung 4.34: Vergleich der Masse von „Modell 1“ und „Modell 2“

Die Werte für die maximale Durchbiegung, welche in Abbildung 4.35 dargestellt sind, unterscheiden sich stark. Bei konstantem l_1 wird mit größer werdendem Winkel der steifste Bereich größer und daher die Durchbiegung kleiner. Bei konstanten l_3 wird jedoch der am

wenigsten steife Bereich länger und daher nimmt die maximale Durchbiegung mit größer werdendem Winkel zu. Das unterschiedliche Verhalten der beiden Modelle ist in Abbildung 4.36 dargestellt.

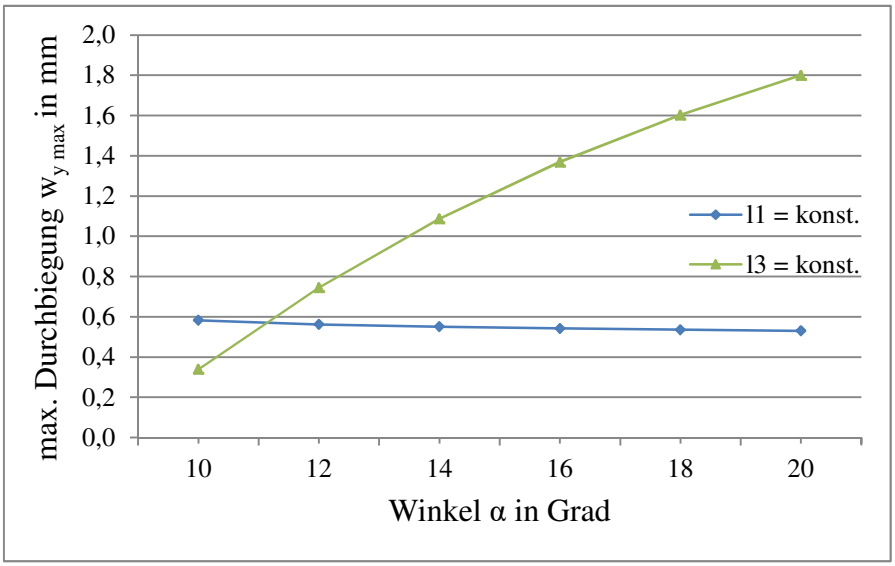


Abbildung 4.35: Vergleich der maximalen Durchbiegung in y-Richtung von „Modell 1“ und „Modell 2“

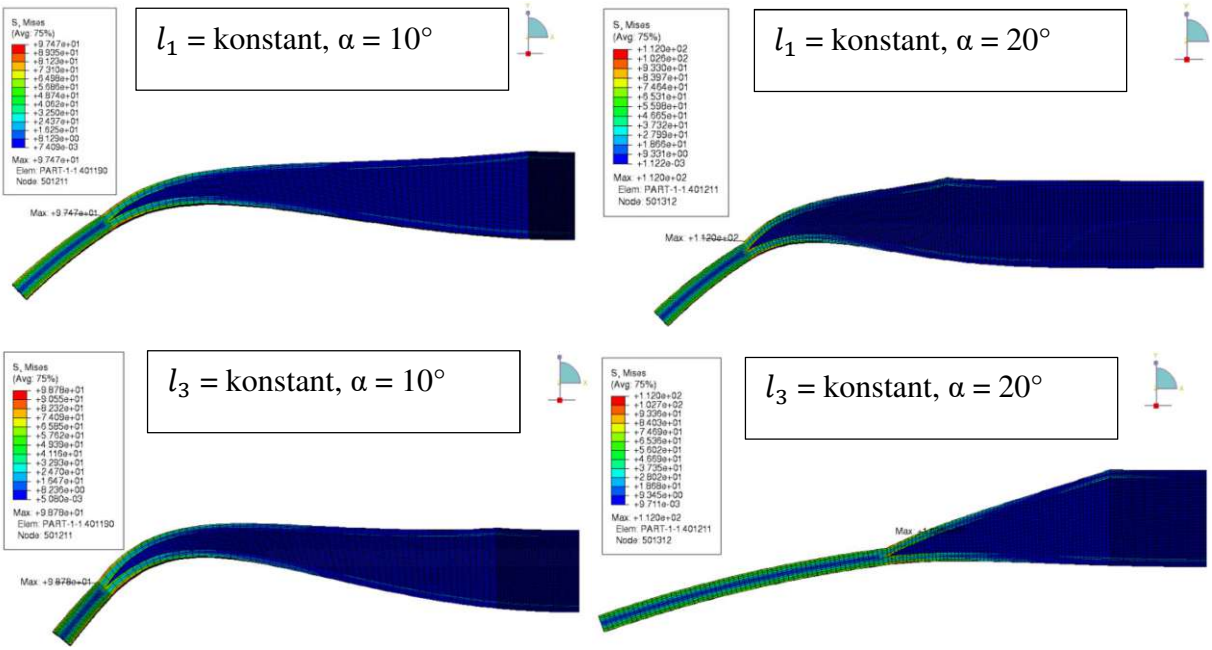


Abbildung 4.36: Gegenüberstellung von Modell 1“ und „Modell 2“

4.2.2 Optimierung mit Dakota

Wie in Kapitel 3.5 beschrieben, müssen die Dateien vom Beispiel der I-Träger Optimierung an das aktuelle Problem angepasst werden. Die Dakota-Input-Datei *Traeger.in*, die *run_iteration.sh* Datei und die Template Dateien müssen an die neuen Variablen α, t_1 und t_2

angepasst werden. Genauso muss auch die Abaqus-Input-Datei des I-Träger für den Übergang modifiziert werden. In der *Zielfunktios.py* Datei muss das Einlesen von der Abaqus Ergebnis Datei verändert werden, so dass die maximale Mises Vergleichsspannung und die Durchbiegung in y-Richtung eingelesen werden. Ferner muss in dieser Datei auch noch die Dichte des Schaums zusätzlich zur Dichte für Aluminium eingegeben werden. Die neuen Variablen, Konstanten und die Formel für die Masse des Übergangs müssen angegeben werden. Hierbei muss berücksichtigt werden, dass sowohl die Abaqus-Input-Datei als auch die Berechnung der Masse sich zwischen Modell 1“ und „Modell 2“unterscheiden.
Als weitere Änderungen in der Dakota-Input-Datei müssen die zulässigen Bereiche für die Variablen eingetragen werden, sowie der Namen der Zielfunktion, die Anzahl der Restriktionen, deren Werte und deren Namen.
Die Daten, mit welchen die Diagramme in diesem Kapitel erzeugt werden, stammen von Parameteranalysen, welche mit Dakota und dem Finite-Elemente-Programm durchgeführt werden.
Zur Optimierung werden die Optimierungsalgorithmen *coliny_cobyla* und *efficient_global* herangezogen, da sie bei der I-Träger Optimierung am besten und schnellsten funktioniert haben.
Untersucht werden vier Fälle, sowohl für das „Modell 1“ als auch für das „Modell 2“. Bei „Fall 1“ soll mit der Variable α die Durchbiegung, bei einer Obergrenze von 110 N/mm^2 für die Mises Spannung, minimiert werden. Das Optimierungsziel und die Variablen von „Fall 2“ sind die gleichen, wie beim „Fall 1“, jedoch ist zusätzlich zur Spannung auch noch die Masse limitiert. Bei „Fall 3“ sind die Restriktionen und das Optimierungsziel gleich wie bei „Fall 1“. Der einzige Unterschied ist, dass es zusätzlich zum Winkel α noch zwei weitere Variablen, die Deckschichtdicken t_1 und t_2, gibt. „Fall 2“ und „Fall 4“ unterscheiden sich, genauso wie „Fall 1“ und „Fall 3“, bei den Variablen.

Fall 1

Der optimale Wert für den Winkel α soll ermittelt werden, sodass die Durchbiegung möglichst klein ist. Zusätzlich sind für die Mises Spannung der konkrete Grenzwert von 110 N/mm^2 gegeben, welcher nicht überschritten werden darf. Die Ergebnisse sind in Tabelle 4.17 zusammengefasst.

Tabelle 4.17: Optimierungsergebnis für Minimierung von der Durchbiegung

kons. Länge	Optimierer	Zielfunktion u. Limit in mm bzw. N/mm^2		Ergebnis für $t_1 = t_2 = 1$ mm		
				$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm^2	α in °
l1	efficient_global	w_y	$\sigma_v \leq 110$	0,538	110,00	17,21
l1	coliny_cobyla	w_y	$\sigma_v \leq 110$	0,538	110,00	17,21
l3	coliny_cobyla	w_y	$\sigma_v \leq 110$	0,339	99,19	10,00
l3	efficient_global	w_y	$\sigma_v \leq 110$	0,339	99,19	10,00

Bei „Modell 1“, wo l_1 konstant ist, sinkt mit steigendem Winkel die Durchbiegung. Die Kurve von der Spannung verhält sich genau umgekehrt. Das zulässige Minimum der Zielfunktion liegt daher beim größtmöglichen Winkel, also beim maximal zulässigsten Wert der Spannung. In Abbildung 4.37 ist dieser Zusamenhang graphisch dargestellt.

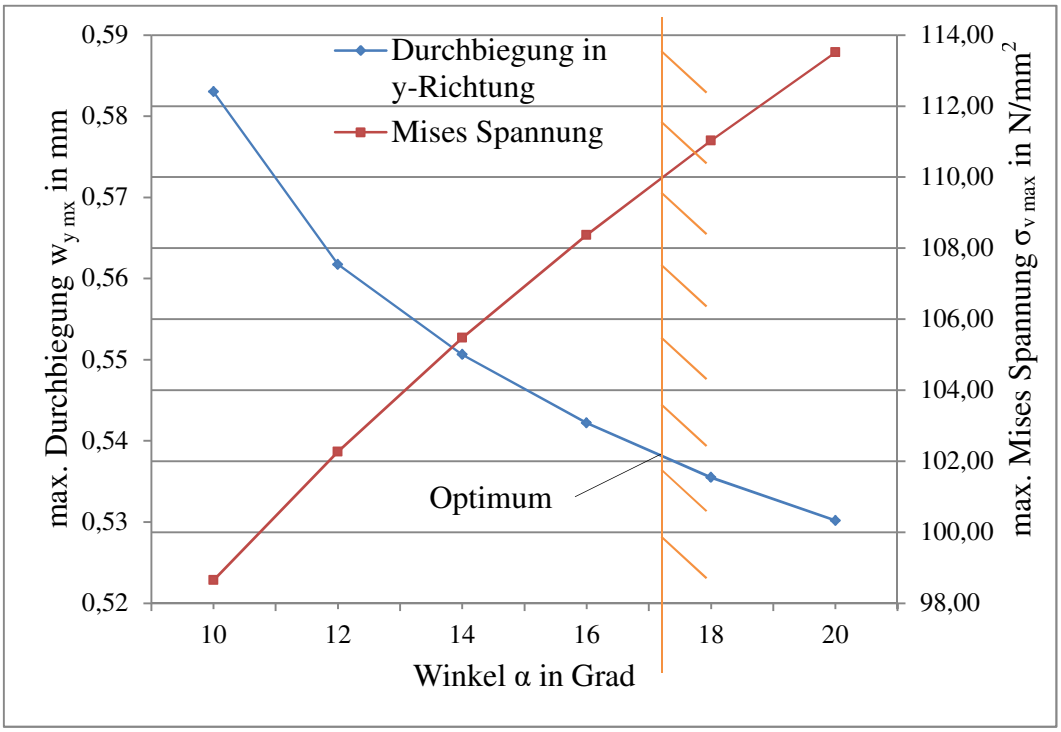


Abbildung 4.37: Fall 1 für „Modell 1“

Beim „Modell 2“, mit konstantem l_3, nimmt sowohl die Spannung als auch die Durchbiegung mit kleiner werdenden Winkel ab. Das Limit der Spannung ist, wie in Abbildung 4.38 ersichtlich, nicht aktiv und das Optimum befindet sich an den Grenzen der Variable α bei 10°.

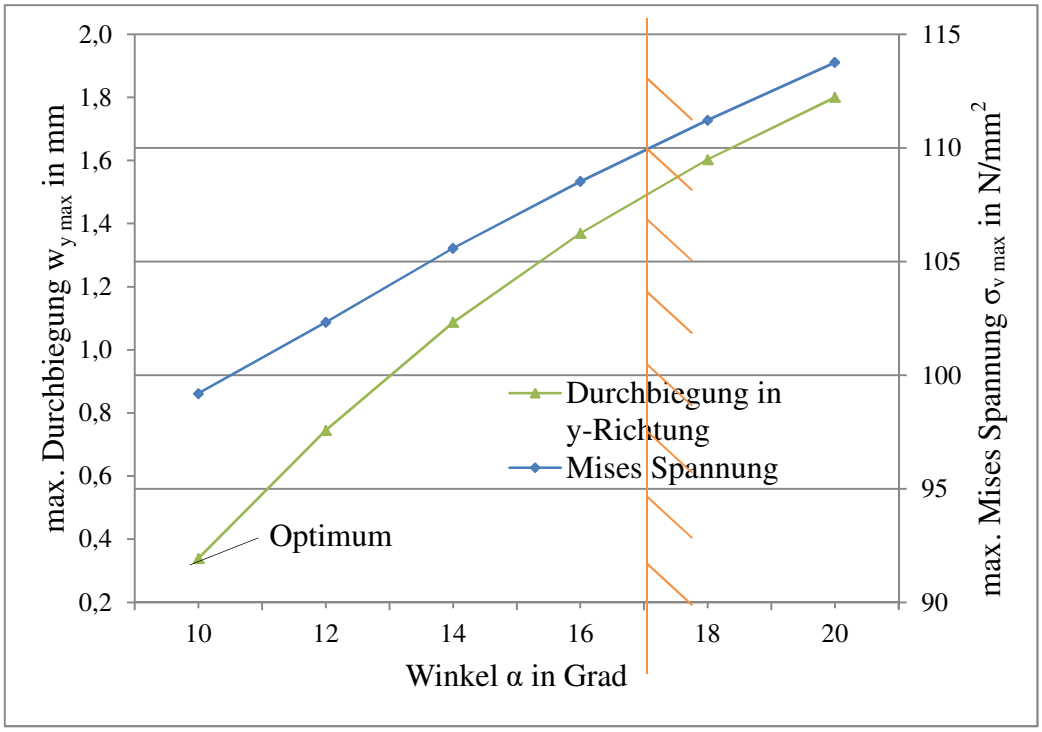


Abbildung 4.38: Fall 1 für „Modell 2“

Fall 2

Der optimale Wert für den Winkel α soll ermittelt werden, so dass die Durchbiegung möglichst klein ist. Zusätzlich sind für die Mises Spannung und die Masse auch noch konkrete Werte gegeben, welche nicht überschritten werden dürfen. Die Ergebnisse sind in Tabelle 4.18 zusammengefasst.

Tabelle 4.18: Optimierungsergebnis für „Fall 2“

kons. Länge	Optimierer	Zielfunktion u. Limit in N/mm² bzw. g		Ergebnis für $t_1 = t_2 = 1$ mm			
				$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	m in g	α in °
l1	efficient_global	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,555	104,21	4,30	13,19
l1	coliny_cobyla	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,555	104,21	4,30	13,19
l3	coliny_cobyla	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,339	99,19	4,292	10,00
l3	efficient_global	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,387	99,19	4,292	10,00

Wie in Abbildung 4.39 dargestellt, schränkt das Limit für die Masse bei „Modell 1“ den zulässigen Bereich mehr ein als das Limit für die Spannung. Da, wie schon bei „Fall 1“ erwähnt, die Durchbiegung mit steigendem Winkel abnimmt, liegt das Optimum, welches in Abbildung 4.40 eingezeichnet ist, beim maximalen Wert der Masse.

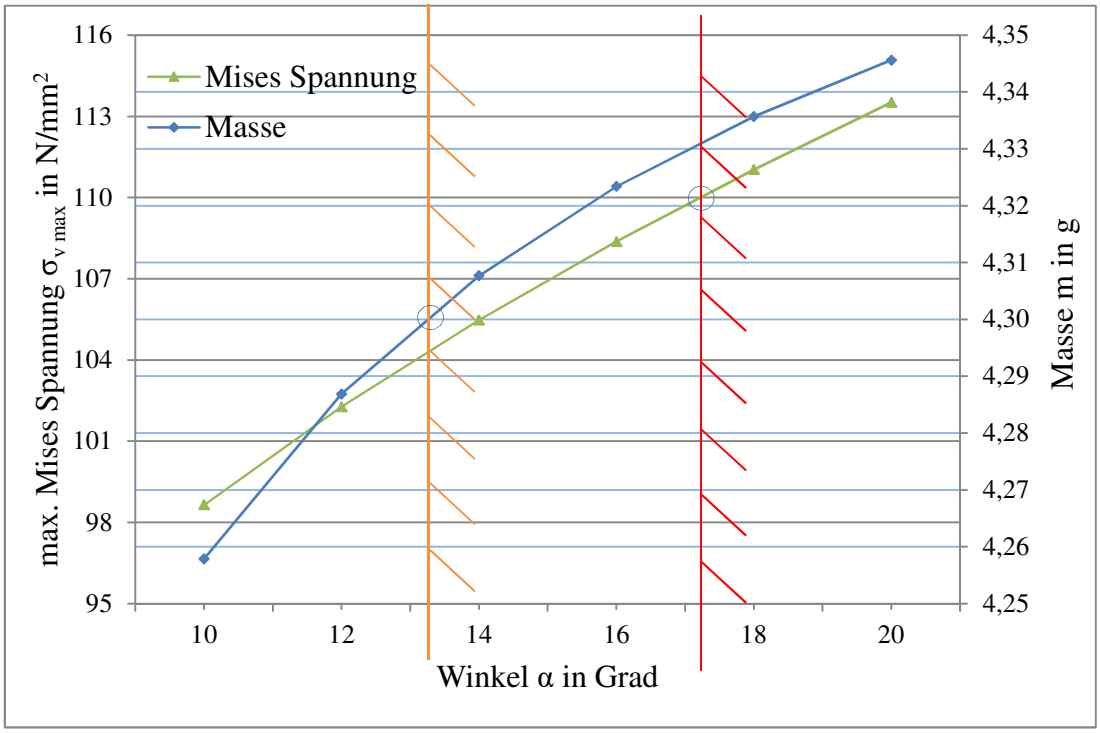


Abbildung 4.39: Limits von „Fall 2“ für das „Modell 1“

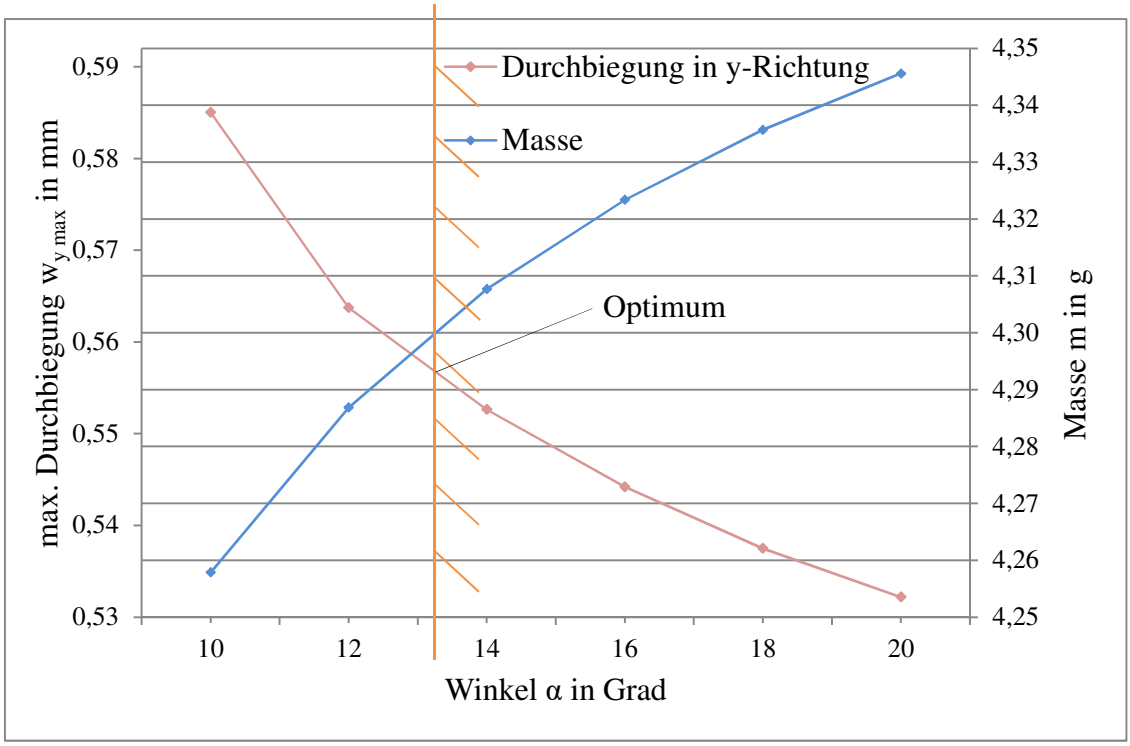


Abbildung 4.40: Optimum von „Fall 2“ für das „Modell 1“

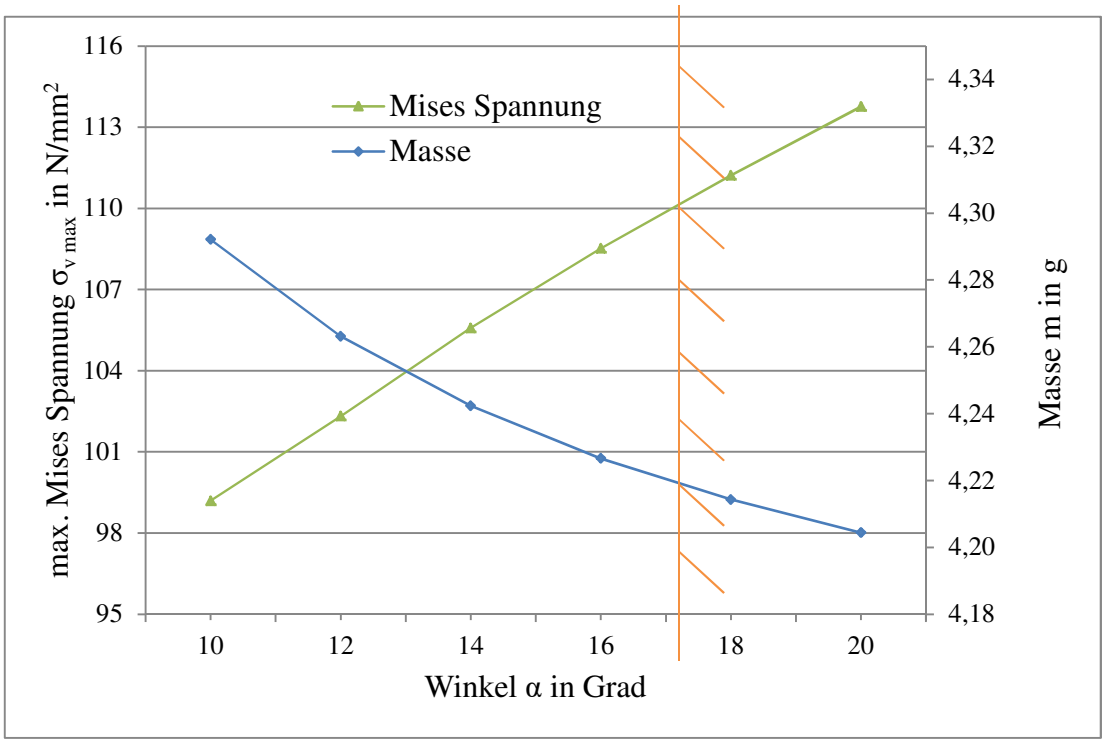


Abbildung 4.41: Limits von „Fall 2“ für das „Modell 2“

Das Limit für die Masse wird bei „Modell 2“ nichtschlagend, da die maximale Masse kleiner ist als das Limit, siehe Abbildung 4.41. Das Ergebnis unterscheidet sich daher nicht von jenem von „Fall 1“ und ist in Abbildung 4.37 dargestellt.

Fall 3

Die optimalen Werte für den Winkel α, die obere Deckschichtdicke t_2 und die unterer Deckschichtdick t_1 sollen ermittelt werden, so dass die Durchbiegung möglichst klein ist. Zusätzlich ist für die Mises Spannung noch eine konkreter Wert gegeben, welcher nicht überschritten werden darf. Die Ergebnisse sind in Tabelle 4.19 zusammengefasst.

Tabelle 4.19: Optimierungsergebnisse für den „Fall 3“

kons. Länge	Optimierer	Zielfunktion u. Limit in N/mm²		Ergebnis				
				$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	α in °	t_1 in mm	t_2 in mm
l1	coliny_cobyla	w_y	$\sigma_v \leq 110$	0,022	12,60	20	3,00	3,00
l1	efficient_global	w_y	$\sigma_v \leq 110$	0,022	12,62	20	3,00	3,00
l3	coliny_cobyla	w_y	$\sigma_v \leq 110$	0,017	10,90	10	3,00	3,00
l3	efficient_global	w_y	$\sigma_v \leq 110$	0,017	10,90	10	3,00	3,00

Wie in Kapitel 4.2.1 erklärt, sinkt die Durchbiegung beim „Modell 1“, wenn α steigt. Wenn t_1 und t_2 zunehmen und dadurch der auf Biegung beanspruchte Querschnitt größer wird, sinkt

die Durchbiegung ebenfalls. Die Spannung nimmt zwar mit steigendem Winkel zu, jedoch ist der Einfluss der Schichtdicken viel größer, welcher die Spannung mit zunehmender Dicke sinken lässt. Durch diesen Einfluss der Dicken sind beim Minimum der Durchbiegung die auftretenden Spannungen sehr viel geringer als das Limit der Spannung. Die minimale Durchbiegung tritt also bei maximalem Winkel und bei maximalen Dicken auf.
Da beim „Modell 2“ mit kleiner werdendem Winkel und zunehmenden Dicken t_1 und t_2 die Spannung und die Durchbiegung abnehmen, ist wie erwartet das Optimum beim Minimum des Winkels und beim Maximum der Dicken.

Fall 4

Der optimalen Werte für den Winkel α, die obere Deckschichtdicke t_2 und die unterer Deckschichtdicke t_1 sollen ermittelt werden, sodass die Durchbiegung möglichst klein ist. Zusätzlich sind für die Mises Spannung und die Durchbiegung auch noch konkrete Werte gegeben, welche nicht überschritten werden dürfen. Die Ergebnisse von der Optimierung sind in der Tabelle 4.20 zusammengefasst.
Beim „Modell 2“ unterscheiden sich die Ergebnisse von „Fall 3“ und „Fall 4“ nicht. Da unabhängig von der Konstellation der Variablen die Masse immer geringer ist als 4,3 kg.
Beim „Modell 1“ ist das Limit für die Masse jedoch aktiv und das Optimum liegt daher nicht, so wie bei „Fall 3“, beim maximalen Winkel, sondern bei 13,19°.

Tabelle 4.20: Optimierungsergebnisse für „Fall 4“

kons. Länge	Optimierer	Zielfunktion u. Limit in N/mm² bzw. g		Ergebnis					
				$w_{y\,max}$ in mm	$\sigma_{v\,max}$ in N/mm²	m in g	α in °	t_1 in mm	t_2 in mm
l1	efficient_global	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,025	11,24	4,30	13,19	3,00	3,00
l1	coliny_cobyla	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,025	11,24	4,30	13,19	3,00	3,00
l3	coliny_cobyla	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,017	10,90	4,26	10,00	3,00	3,00
l3	efficient_global	w_y	$\sigma_v \leq 110$, $m \leq 4{,}3$	0,017	10,90	4,26	10,00	3,00	3,00

5 Schlussfolgerung

Die einfache Geometrie des I-Trägers hat gezeigt, dass das Auffinden einer analytischen Optimierungslösung nur für wenige Spezialfälle möglich ist. Denn unabhängig davon, welche der drei möglichen Optimierungsvariablen man für eine Optimierung mit zwei Variablen verwendet, gibt es keine analytische Lösung. Es ist nur eine analytische Lösung auffindbar, wenn das Problem auf eine Variable reduziert wird.

Bei der Optimierung des I-Trägers mit nur einer Variable, haben alle verwendeten Optimierer die Extremstelle gefunden, bei zwei Variablen hat nur noch einer exakt das Optimum gefunden. Zwei weitere Optimierer haben einen Wert in der Nähe des Optimums errechnet und zwei andere kamen zu einem anderen Ergebnis. Bei der numerischen Optimierung nimmt die Komplexität der Optimierung also mit steigender Variablenanzahl rasch zu.

Es lässt sich auch mit geringem Aufwand das vorhandene Optimierungsprogramm für eine andere Geometrie adaptieren. Neben der Anpassung der Dakota Dateien an die jeweilige Fragestellung, wird dafür auch ein parametergesteuertes Finite-Elemente-Modell der Geometrie benötigt. Die größte Herausforderung ist die Wahl eines geeigneten Optimierungs-algorithmus. Generell kann man sagen, dass bei den durchgeführten Optimierungen die beiden Optimieralgorithmen *coliny_cobyla* und *efficient_global* gute Ergebnisse geliefert haben. *coliny_cobyla* ist im Gegensatz zu den globalen Optimierer *efficient_global* ein lokaler. Da unsere beiden Beispiele eine einfache Geometrie aufweisen, sind die bei den Beispielen verwendeten Zielfunktionen konvex und deshalb findet auch der lokale Optimieralgorithmus *coliny_cobyla* das Optimum, und zwar um einiges schneller als die meisten andern. Globale Optimierer erreichen meistens rasch den Bereich, in welchem das Optimum liegt, jedoch dauert es dann meistens recht lange bis sie das genaue Optimum gefunden haben. Es bietet sich daher an, den Bereich für das Optimum mit den globalen Verfahren zu bestimmen und dann mit einen lokalem Optimierer das genaue Optimum zu suchen.

Die zwei Optimierungsbeispiele, haben also gezeigt, dass eine Formoptimierung mit den Optimierungs-Programm Dakota und den Finite-Elemente-Programm Abaqus realisierbar ist.

6 Literaturverzeichnis:

[1] Brain M. Adams, Michael S.Eldred, Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.11 User's Manual, 2019 Sandia National Labortories, P.O. Box 5800, Albuquerqe, New Mexico 87185

[2] Kurt Machinek, Optimaldimensionierung, Galgasse 25/4, 1130 Wien

[3] Airex AG, Technical Data Sheet Airex R82, Industrie Nord 26, 5643 Sins, Swizerland, https://www.3accorematerials.com/uploads/documents/TDS-AIREX-R82-D-07.2011_1106.pdf, (27.4.2022)

[4] F.G.Rammerstorfer, Einführung in die Finite Elemente Methode, 2014, TU-Wien

[5] Powell M.D.J., A direct search optimization method that models the objective function and constraint functions by linear interpolation. Advances in optimization and numerical analysis: proceedings of the Sixth Workshop on Optimization and Numerical Analysis Hrsg. Von S. Gomez & J.-P.Hennart, S. 51-67, 1994, Cambridge

[6] Donald R.Jones, Joaquim,R,R,A Martins, The DIRECT algorithm: 25 years Later, Department of Aerospace Engineering, 2019, University of Michigan, Ann Arbor, MI, USA

[7] Steffen Kux, Hybride Optimierungsstrategien für komplexe technische Aufgabenstellungen, 2011,Mittweida

[8] Martin Ströcker, Untersuchung von Optimierungsverfahren für rechenzeitaufwendige technische Anwendungen in der Motorentwicklung, 2007, Chemniz

[9] David E. Goldberg, Generic Algorithms, 1989, Alabama

[10] F.G.Rammerstorfer, Skript zur Vorlesung Leichtbau, 2015, TU Wien

[11] M. Alexander Roth, IVW – Schriftenreihe Band 61, 2006, TU Kaiserslautern

[12] Alois Starliner, Vorlesungsunterlagen Sandwich Structures, 2019, TU Wien

[13] Brian M. Adams, Lara E. Bauman, Dakota Reference Manual, Large-Scale Engineering Optimization and Uncertainty Analysis Version 6.0, 2014, https://dakota.sandia.gov/sites/default/files/docs/6.0/html-ref/index.html, (16.10.2022)

[14] Dassault Systèmes, 2014, Abaqus Analysis User's Guide 6.14, http://130.149.89.49:2080/v6.14/books/usb/default.htm, (16.10.2022)

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Abkürzungsverzeichnis

EGO ..... Efficient Global Optimization
COBYLA ..... Constrained Optimization BY Linear Approximations
DIRECT ..... Dividing Rectangles
NKST ..... Nachkommerstellen
KC ..... Kinematic Coupling
E ..... Equations
kons. ..... Konstante

A. Anhang

A.1 Dakota-Input-Datei

In der folgenden Abbildung ist die Dakota-Input-Datei dargestellt, welche den Ablauf der Optimierung festlegt.

```
environment
  tabular_data
    tabular_data_file  'Traeger.dat'
 ##
method
   coliny_cobyla
   max_iterations = 500
   max_function_evaluations = 500
 ##
model
   single
 ##
variables
  continuous_design  2
     lower_bounds          1        1
     upper_bounds          14.5    19
     descriptors           "z"      "y"
 ##
interface
    analysis_driver 'run_iteration.sh'
    fork
    parameters_file 'parameters.dak'
    results_file 'results.dak'
    work_directory
      named = 'iteration'
      copy_files = 'Job/ParInput_template.inp' 'Job/run_iteration.sh' 'Job/Traeger.inp'
'Job/parameters_template.sh' 'Job/Zielfunktion.py'
      directory_tag
      directory_save
      replace
   file_save
   file_tag
 ##
responses
   descriptors "A*w"  "w" "S11"
   objective_functions  = 1
   nonlinear_inequality_constraints 2
       nonlinear_inequality_upper_bounds 2.2  400
   no_gradients
   no_hessians
```

Abbildung A. 1: Dakota-Input-Datei, Traeger.in

A.2 **Einlesevarianten**

Dieser Abschnitt beinhaltet die drei Versionen von den *analysis_driver run_iteration.sh* und der Datei *Zielfunktion.py*, welche in dieser Arbeit verwendet werden. Die drei Varianten heißen: „Text 1“, „Text 2“ und „odb“.

Text 1

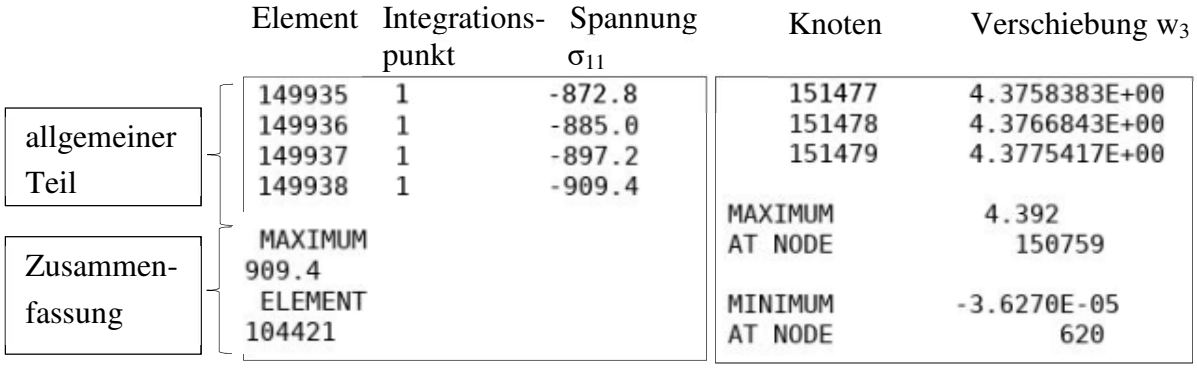


Abbildung A. 2: Auszug aus der Text-Datei *Abaqus_Ergebnis.dat*

```
#!/usr/bin/env bash

../Job/dprepro.perl $1 ParInput_template.inp ParInput.inp
../Job/dprepro.perl $1 parameters_template.sh parameters.sh

/programs/shared/hks/Commands/abq2020 interactive job=Abaqus_Ergebnis
input=Traeger.inp cpus=4

grep 'MAXIMUM  ' Abaqus_Ergebnis.dat | cut -c 10- > results.tmp
sed -i -re 's#[[:space:]]+##g;/^$/d' results.tmp
. parameters.sh
Zielfunktion.py $z $y

mv results.tmp  $2
```

Schritt 1

Schritt 2

Schritt 3

Schritt 4

Abbildung A. 3: *analysis_driver, run_iteration.sh*, Version „Text 1”

```
#!/usr/bin/env python

fa = open('results.tmp', 'r')
a=[]
for line in fa:
    line = line.rstrip()
    b=float(line)
    a.extend([b])

max_s = a[0]
max_w = a[1]

import sys
z = float(sys.argv[1])
y = float(sys.argv[2])
h = 30.
b = 20.
t=2*b*z+y*(h-2*z)*max_w

with open('results.tmp', 'w') as ofile:
  ofile.write("%.11f\n%.11f\n%.11f\n " %(t, max_w, max_s))
```

Abbildung A. 4: *Zielfunktion.py*, Version „Text 1“

Text 2

```
#!/usr/bin/env bash

../Job/dprepro.perl $1 ParInput_template.inp ParInput.inp
../Job/dprepro.perl $1 parameters_template.sh parameters.sh

/programs/shared/hks/Commands/abq2020 interactive
job=Abaqus_Ergebnis input=Traeger.inp cpus=4

grep 'MAXIMUM  ' Abaqus_Ergebnis.dat | cut -c 10- > results.tmp
sed -i -re 's#[[:space:]]+##g;/^$/d' results.tmp
grep '150759  ' Abaqus_Ergebnis.dat | cut -c 12- > results2.tmp
sed -i -re 's#[[:space:]]+##g;/^$/d' results2.tmp
. parameters.sh
Zielfunktion.py $z $y

mv results.tmp  $2
```

Schritt 1

Schritt 2

Schritt 3

Schritt 4

Abbildung A. 5: *analysis_driver, run_iteration.sh*, Version „Text 2”

```
#!/usr/bin/env python

fa = open('results.tmp', 'r')
a=[]
for line in fa:
    line = line.rstrip()
    b=float(line)
    a.extend([b])

fat = open('results2.tmp', 'r')
c=[]
for line in fat:
    line = line.rstrip()
    b=float(line)
    c.extend([b])

if abs(a[1]-c[0])<0.001 :
   a[1]=c[0]
max_s = a[0]
max_w = a[1]

import sys
z = float(sys.argv[1])
y = float(sys.argv[2])
h = 30.
b = 20.
t=2*b*z+y*(h-2*z)* max_w

with open('results.tmp', 'w') as ofile:
ofile.write("%.11f\n%.11f\n%.11f\n " %(t, max_w, max_s))
```

Abbildung A. 6: *Zielfunktion.py*, Version „Text 2"

Odb

```
#!/usr/bin/env bash

../Job/dprepro.perl $1 ParInput_template.inp ParInput.inp
../Job/dprepro.perl $1 parameters_template.sh parameters.sh

/programs/shared/hks/Commands/abq2020 interactive job=Abaqus_Ergebnis
input=Traeger.inp cpus=4

. parameters.sh
Zielfunktion.py $z $y

mv results.tmp  $2
```

Schritt 1

Schritt 2

Schritt 3

Schritt 4

Abbildung A. 7: analysis_*driver, run_iteration.sh*, Version „odb"

```
#!/programs/shared/hks/Commands/abq2020 python
from odbAccess import *

odb = openOdb(path='Abaqus_Ergebnis.odb')
center_dis = odb.rootAssembly.instances['PART-1-1']
.nodeSets['TRAEGERN']
Displacement = odb.steps['Step-1'].frames[-1].
fieldOutputs['U'].getSubset(region=center_dis)
Values_dis = Displacement.values
a_w=[]
for v in Values_dis:
    b_w=float(v.data[2])
    a_w.extend([b_w])
max_w= max(a_w)

center_stress = odb.rootAssembly.instances['PART-1-1'].
elementSets['TRAEGERE']
stress_Field = odb.steps['Step-1'].frames[-1].fieldOutputs['S'].
getSubset(region=center_stress, position=INTEGRATION_POINT,
 elementType = 'C3D8R')
Values_stress = stress_Field.values
a=[]
for v in Values_stress:
            b=float(v.data[0])
            a.extend([b])
max_s= max(a)

import sys
y = float(sys.argv[2])
z = float(sys.argv[1])
h = 30.
b = 20.
f = 2*b*z+y*(h-2*z)
t=f*max_w
with open('results.tmp', 'w') as ofile:
  ofile.write("%15.11f \n%15.11f \n%15.11f " %(t, max_w, max_s))
```

Abbildung A. 8: *Zielfunktion.py*, Version „odb“

A.3 Abaqus-Input-Dateien für den I-Träger

```
*PARAMETER
z=2.4
y=1.
i=1
```

Abbildung A. 9: *ParInput.inp*

```
*HEADING
I-Traeger unter reiner Biegung mit parametrisierter Geometrie
Querschnittsoptimierung
*******Parameterdefinition**********************
*INCLUDE, INPUT=ParInput.inp
**Werte in mm
*PARAMETER
h=30.
b=20.
r= 0.2
l=100.
ls=10.
ls1=l+ls
hmz=h-z
hmr=h-r
bmy2=(b/2-y/2)
b2=b/2
h2=h/2
******  Knotendefinition*****************************
******* Traeger********
*NODE
1, 0., 0., 0.
2, 0., 0., <r>
6, 0., 0., <z>
34, 0., 0., <hmz>
38, 0., 0., <hmr>
39, 0., 0., <h>
*NGEN, NSET=l1
2, 6, 1
34, 38, 1
*NSET, NSET=L1
l1, 1, 39
*NCOPY, OLD SET=L1, CHANGE NUMBER=600, NEW SET=L2, SHIFT
0, <bmy2>, 0
0, 0, 0,    0, 1, 0,    0
*NFILL, NSET=a1
L1, L2, 15, 40
*NGEN, NSET=l3
606, 634, 1
*NSET, NSET=L3
l3, L2
*NCOPY, OLD SET=L3, CHANGE NUMBER=240, NEW SET=L4, SHIFT
0, <y>, 0
0, 0, 0,    0, 1, 0,    0
*NFILL, NSET=a2
L3, L4, 6, 40
*NCOPY, OLD SET=L2, CHANGE NUMBER=240, NEW SET=L5, SHIFT
0, <y>, 0
0, 0, 0,    0, 1, 0,    0
*NCOPY, OLD SET=L5, CHANGE NUMBER=600, NEW SET=L6, SHIFT
```

```
0, <bmy2>, 0
0, 0, 0,    0, 1, 0,    0
*NFILL, NSET=a3
L5, L6, 15, 40
*NSET, NSET=A0
a1, a2, a3
*NCOPY, OLD SET=A0, CHANGE NUMBER=150000, NEW SET=AL, SHIFT
<l>, 0, 0
0, 0, 0,    0, 1, 0,    0
*NFILL, NSET=TraegerN
A0, AL, 100, 1500
**********Balken*******
*Node
161370, <ls1>, <b2>, <h2>
160170, <l>, <b2>, <h2>
*NGEN, NSET=StabN
160170, 161370 , 400
*NSET, NSET=AlleN
StabN, TraegerN
******  Elementedefinition************************
******* Traeger******
*ELEMENT, TYPE=C3D8R
1, 1, 1501, 1541, 41, 2, 1502, 1542, 42
*ELGEN, ELSET = A1
1, 100, 1500, 1500, 36, 40, 40, 5, 1, 1
*ELEMENT, TYPE=C3D8R
606, 606, 2106, 2146, 646, 607, 2107, 2147, 647,
*ELGEN, ELSET = A2
606, 100, 1500, 1500, 6, 40, 40, 28, 1, 1
*ELEMENT, TYPE=C3D8R
34, 34, 1534, 1574, 74, 35, 1535, 1575,75
*ELGEN, ELSET = A3
34, 100, 1500, 1500, 36, 40, 40, 5, 1, 1
*ELSET, ELSET= TraegerE
A1, A2,A3
*********Balken**********
*ELEMENT, TYPE=B33
160000, 160170, 160570
*ELGEN, ELSET = center-beam
160000, 3, 400, 1
*ELSET,ELSET=AlleE
TraegerE, center-beam
**** Materialdefinition**************************
******* Traeger***********
*MATERIAL, NAME=ALU
*ELASTIC, TYPE = ISOTROPIC
70000.0,0.33
*MATERIAL, NAME=STIFF-BEAM
*ELASTIC, TYPE = ISOTROPIC
70000000.0    ,0.33
*********Balken********
*SOLID SECTION, ELSET=TraegerE , MATERIAL=ALU
*BEAM SECTION, ELSET=center-beam, MATERIAL= STIFF-BEAM, SECTION=I
15,30,20,20,2.26,2.26,1
****  Verbindung von Balken und Traeger************
*KINEMATIC COUPLING, REF NODE=160170
AL,1,1
AL,4,6
150740, 2,3
```

```
************** Randbedingungen und Ausgabe*************
*STEP, PERTURBATION
Test
*STATIC
*BOUNDARY, TYPE=DISPLACEMENT
A0,1,,0.
740,2,,0.
740,3,,0.
721,2,,0.
*CLOAD
160570, 5, -625000.
*FILE FORMAT, ASCII
*OUTPUT, FIELD
*NODE OUTPUT
U,CF,RF
*ELEMENT OUTPUT
S,E,IVOL
*EL PRINT,ELSET=TraegerE
S11
*NODE PRINT,NSET=TraegerN
U3
*EL FILE, ELSET=AlleE
S
*NODE FILE, NSET=AlleN
U
*END STEP
```

Abbildung A. 10:*Traeger.inp*