Entwicklung einer algorithmischen Parameteroptimierung der mechanischen Strukturauslegung mit einem kommerziellen FE-Code

Der Fakultät für Maschinenbau
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte Dissertation

von
Dipl.-Ing. Markus Riem
geboren am 13. November 1969
in Stuttgart

2011

Tag der mündlichen Prüfung: 09.05.2011

1. Referent: Prof. Dr.-Ing. D. Besdo

2. Referent: Prof. Dr.-Ing. G.Poll

Vorsitz: Prof. Dr.-Ing. B.-A. Behrens

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Maschinenbau der Fachhochschule Flensburg in Zusammenarbeit mit dem Institut für Kontinuumsmechanik der Universität Hannover.

Dem ehemaligen Leiter des Instituts für Kontinuumsmechanik, Herrn Prof. Dr.-Ing. Dieter Besdo, gilt mein besonderer Dank für die Anregungen zu dieser Arbeit und die großzügige Unterstützung bei der Durchführung. Herrn Prof. Dr.-Ing. Gerhard Poll, dem Leiter des Instituts für Maschinenelemente, Konstruktionstechnik und Tribologie, danke ich für die bereitwillige Durchsicht der Arbeit und die sich daraus ergebenden fachlichen Hinweise. Ebenso danke ich Herrn Prof. Dr.-Ing. Bernd-Arno Behrens, dem Leiter des Instituts für Umformtechnik, für die Übernahme des Prüfungsvorsitzes.

Besonderer Dank gilt auch Herrn Prof. Dr.-Ing. Axel Krapoth und Prof. Dr.-Ing. Ernst Reimers von der Fachhochschule Flensburg für deren großzügige Unterstützung, ohne die die Arbeit nicht möglich gewesen wäre.

Meinen Eltern danke ich für die Unterstützung und Förderung, die sie mir während der Zeit haben zukommen lassen. Mein ganz besonderer Dank gilt meiner Frau Maike für ihre Geduld und ihre Unterstützung während der Entstehung der Arbeit.

Flensburg, im Mai 2011

Markus Riem

Inhaltsverzeichnis

Formelzeichen

Lateinische Notation

A_D	Oberfläche der Dose
a_i	Querschnittsfläche des Stabe i
B	konstantes Volumen des Balkens
$\vec{b}_a$	Projektion eines Vektors $\vec{b}$ auf einen Vektors $\vec{a}$
b_i	jeweilige Breite des Balkenabschnitts
$\vec{F}$	Äußere Kräfte
$f(\vec{x})$	Zielfunktion
$g_j(\vec{x})$	Ungleichheitsrestriktion
h	einflussreichste Schrittweite eines Tastzyklus
$h_k(\vec{x})$	Gleichheitsrestriktion
$\overline{K}(x)$	Steifigkeitsmatrix
$L(\vec{x};\vec{\lambda})$	Lagrangsche-Funktion für restringierte Optimierungsprobleme
l_i	Länge des Stabes i
q	Streckenlast
r	Faktor der Straffunktion
$\vec{S}$	Ausführungsschritt:
$\vec{u}$	Knotenverschiebungen
V	Anzahl der Variablen
$\vec{v}$	Volumengradient
V_D	Volumen der Dose
V_S	Volumen des Stabdreischlags
w_0	Absenkung des Balkens an der Stelle 0
x^k	Startwert

x^{k+1}	Neue Variablenwerte
z_r	Zielfunktion

Griechische Notation

ρ	spezifisches Gewicht
Φ	Potential: Änderung des Volumens pro Änderung der Spannung
Δx^k	Schrittweiten
σ_{zul}	zulässige Spannung
Δd	Startschrittweite
μ_j^2	Schlupfvariable der Lagrangsche-Funktion

Kurzfassung

Häufig muss im technischen Entwicklungsprozess die optimale Lösung ermittelt werden, um die gestellten Zielvorgaben zu erfüllen. Dies kann ein langwieriger und schwer durchzuführender Prozess sein.

In dieser Arbeit wurde für den Bereich der Strukturmechanik eine automatisierte Optimierungsumgebung auf der Basis des kommerziellen FEM-Programms Abaqus entwickelt. Mit ihr ist es möglich, einzelne FEM-Rechnungen sinnvoll nacheinander von der Optimierung ausführen zu lassen, bis die gestellten Zielvorgaben erreicht sind.

Dazu wurde die grafische Oberfläche Abaqus/CAE so erweitert, dass der Anwender durch die Optimierung geführt werden kann. Es öffnen sich die entsprechenden Fenster, in denen die Parameter der Optimierung, wie Startwerte und Schrittweiten eingegeben werden können und danach die Optimierung gestartet werden kann. In einer Lernphase der Optimierung definiert der Anwender die zu variierenden Parameter. Dazu wird ein vorher erstelltes FEM-Modell des beschriebenen Problems benötigt. In diesem Modell (Abaqus/CAE) können sämtliche Längen und die Form des Modells durch die Wahl der Parameter, die keiner Beschränkung unterliegen, geändert werden. Durch diese beispielhafte Änderung werden die Parameter für die Optimierung definiert.

Gesteuert wird die Durchführung der Optimierung durch Algorithmen. Diese sind der Kern der Optimierung und wurden extra für den spezifischen Bereich der Strukturoptimierung entwickelt. Dabei handelt es sich um sukzessive Suchverfahren und um Gradientenverfahren erster Ordnung. Die Ziel- oder Gütefunktion kann der Verlauf der Kontaktkräfte oder der Verschiebung einzelner Punkte sein, oder das restringierte Problem der Gewichtsoptimierung mit der Nebenbedingung der zulässigen Spannung. Die Suchverfahren laufen am stabilsten und sie sind für alle Probleme und Problemarten geeignet. Sie sind allerdings langsamer. Die Gradientenverfahren dagegen sind in der Lage schnell zum Ziel zu kommen, können jedoch, z.B. bei Unstetigkeiten, Probleme bekommen den Gradienten zu ermitteln.

Schlagworte

Strukturoptimierung, Finite Elemente, Algorithmen

Abstract

Often must be determined in the technical development process, the optimal solution to meet the set targets. This can be a lengthy and difficult process to be undertaken.

This work was developed for the field of structural mechanics, an automated optimization environment on the basis of the commercial FEM-program Abaqus. It makes it possible to have run individual FEM calculations useful in succession from the optimization until the identified targets are achieved.

For this, the graphical interface Abaqus / CAE has been extended so that the user can be guided by the optimization. It will open the appropriate window, in which the parameters of the optimization as starting values and step sizes can be entered and then the optimization can be started. In the learning phase of optimization, the user defines the parameters to be varied. This needs a created FEM model of the described problem. In this model (Abaqus / CAE), all lengths and the shape of the model by the choice of the parameters that are not subject to restriction can be changed. Through this example of change the parameters for optimization are defined.

The execution of the optimization is controlled through the implementation of optimization algorithms. These are the core of the optimization and are specially designed for the specific area of structural optimization. This is to gradually search methods and gradient methods for first order. The goal or objective function may be the course of the contact forces and the displacement of individual points, or the restricted problem of weight optimization with the constraint of the allowable stress. The search procedures are most stable and are suitable for all problems and problem types. However, they are slower. The gradient methods, however, are able to come quickly to the target, but may, for example, at discontinuities, get problems to get the gradient to be determined.

Keywords

Structural optimization, finite elements, algorithms

1 Einleitung

Bei der Entwicklung und der Konstruktion in der Technik geht es in der Regel darum, eine gestellte und definierte Aufgabe umzusetzen. Dies ist, je nach dem Schwierigkeitsgrad der Aufgabe oder der Erfahrung des Ingenieurs, sehr leicht oder sogar fast trivial in der Umsetzung oder wird einfach routinemäßig abgearbeitet. Meistens ist aber die zu lösende Aufgabe sehr viel komplexer. Dann wird, wie von neueren Konzepten der Konstruktionsmethodik vorgeschlagen, weniger nach der Intuition des Ingenieurs gefragt, als vielmehr die Problembearbeitung in Phasen unterteilt und diese getrennt voneinander behandelt. Diese Unterscheidung könnte bedeuten, dass zuerst die Aufgabe definiert wird. Weiterhin wird konzipiert, wobei mehrere unterschiedliche prinzipielle Lösungsmöglichkeiten erarbeitet und durch orientierende Berechnungen und Versuche gestützt werden. Aus dem besten Konzept wird ein maßstäblicher Entwurf erstellt und in ihm werden Gestaltungszonen definiert, die noch verändert und variiert werden können. Darauf folgt das Ausarbeiten, das Gestalten der Einzelteile, Erstellen der Ausführungsunterlagen und Herstellen eines Prototyps. In diesem Ablauf der Konstruktion kommt es häufig vor, dass ein Problem definiert wurde, an dessen Lösung einige Forderungen gestellt wurden, so dass es schwierig ist ohne weiteres eine Lösung zu erarbeiten. Oft ist es der Fall, dass ein Bauteil zwar bestimmte Belastungen ertragen muss, dabei aber nicht zu schwer sein darf, um andere Bauteile nicht zu sehr zu belasten. Dann kann es sein, dass eine erste Lösungsmöglichkeit nicht ausreicht, sondern, dass sie noch verbessert werden muss. In einem solchen Fall benötigt man eine Optimierung.

Es gibt prinzipiell zwei mögliche Anlässe für eine Optimierung. Entweder ist die Funktionalität einer Konstruktion erst ab einer gewissen Güte der Entwicklung gegeben oder man hat sich das Ziel gesetzt, so gut wie möglich zu sein. In beiden Fällen ist man bestrebt, seine Entwicklung bis zu einem gewissen Grad zu verbessern. Im Falle einer Optimierung wird das Optimum, also die beste Möglichkeit gesucht. Die Entscheidung für die Notwendigkeit einer Optimierung ist eine einfache Sache, ihre Durchführung kann jedoch mit Problemen verbunden sein. Es kann durchaus sein, dass ihre Durchführung kein Problem ist, die Lösung recht klar auf der Hand liegt und mit einigen wenigen Arbeitsschritten vom Konstrukteur oder vom Berechner erarbeitet werden kann. Meistens ist das aber nicht so einfach. Bei einem klassischen Optimierungsproblem bedingen sich die Einflussgrößen gegenseitig, wodurch beim Verändern einer Größe die Entwicklung einer anderen nicht vorhergesagt werden kann, und dazu hat man meistens mehrere Einflussgrößen. Bei

solchen Problemen verliert der Mensch schnell den Überblick und findet keine Lösung oder erst nach sehr langer Zeit. So etwas bindet den Ingenieur zu lange. Deshalb liegt der Schluss nahe, auch diese Arbeit von Maschinen, in diesem Fall von Rechenmaschinen übernehmen zu lassen, um den Menschen zu entlasten oder die Lösungsgüte gegebenenfalls zu erhöhen.

Optimierungsrechnungen lassen sich in jeder Phase der beschriebenen Konstruktionsmethodik durchführen. In der Konzeptionsphase wurden orientierende Berechnungen und Versuche erwähnt. Schon diese Mittel könnten genutzt werden, um Einflussgrößen zu ermitteln und zu bewerten und daraus eine Optimierung zu entwerfen. Voraussetzungen für eine Optimierung sind entweder ein mathematisch-physikalisches Modell, das berechnet und gelöst werden kann, um so aus den Ergebnissen Schlüsse zu ziehen und so das Optimum zu finden, oder dass diese Ergebnisse durch Versuche festgestellt wurden. Welche prinzipiellen Formen der Optimierung es gibt, wird in diesem Kapitel behandelt werden, aber neben der analytischen Lösung, die nur einen sehr kleinen Platz bei dem Thema der Optimierung einnimmt, werden die meisten Optimierungen mit dem Computer durchgeführt, was, wie schon gesagt, meistens ein mathematisch-physikalisches Modell des Problems benötigt und auf der anderen Seite eine programmtechnische Umsetzung der Optimierung und der Algorithmen aus denen sie besteht. Algorithmen sind sich wiederholende Ausführungsanweisungen und deshalb für die Umsetzung durch den Computer geeignet. Für dieses Stadium der Entwicklung kann man selber eine Optimierung programmieren oder man nutzt einfache Optimierungs-umgebungen, wie es sie z.B. in dem Mathematikprogramm Matlab gibt.

Diese Arbeit jedoch befasst sich mit der Strukturoptimierung mechanischer Strukturen und ist damit in den Phasen Entwurf und Ausarbeitung angesiedelt [Hör04, Scwa01]. Damit ist man in dem Bereich, in dem schon recht genaue mechanische Modelle der Konstruktionen bestehen. In diesen Modellen, lassen sich nun die Einflussgrößen sehr genau definieren und so die Variablen des Optimierungsproblems festlegen [Fis03, Fis04]. Dazu ist zu sagen, dass die recht genauen mechanischen Modelle meistens in Form von Finite-Element-Methode-Modellen (FEM-Modelle) vorliegen und es gut wäre, diese auch für die Optimierung zu nutzen. An diesem Punkt setzt diese Arbeit an. Hier soll eine Optimierungsumgebung für den FEM-Anwender entwickelt werden, mit der es möglich ist, die mit FEM modellierten Systeme zu optimieren.

Zum Verständnis der Grundlagen der Optimierung, werden diese in diesem Kapitel behandelt werden [Pap91, Lit92]. Dabei wird sehr schnell der Focus auf die Strukturoptimierung gerichtet und weiter zur Spezialisierung der Strukturmechanik, die Optimierung mechanischer Strukturen mit der FEM, die auch das Thema dieser Arbeit ist [Scwe75, Hil88, Els78]. Im letzten Abschnitt des Einleitungskapitels wird das Ziel dieser Arbeit beschrieben werden.

1.1 Grundlegende Optimierungsarten

Bei der Optimierung geht es darum das Optimum zu finden. Das bedeutet, dass der Ergebniswert einer Rechnung, je nach Definition, so klein oder so groß wie möglich sein soll. Um eine Rechnung durchführen zu können, ist es notwendig, dass ein berechenbares Modell vorliegt. Das ist im Fall dieser Arbeit ein mechanisches Modell einer Struktur, also ein Strukturmodell. Der zugehörige Ergebniswert könnte zum Beispiel die Verschiebung eines Punktes bei Belastung, das Gewicht der Struktur oder seine erste Eigenfrequenz sein. Um den Ergebniswert zu verkleinern, muss an der Struktur irgendetwas geändert werden und dazu müssen Variablen definiert werden, die veränderbare Größen der Struktur sind. Der Ergebniswert, der geändert werden soll, wird die *Zielfunktion* der Optimierung genannt.

Ein großer Bereich der Optimierung ist die Strukturoptimierung, die Optimierung mechanischer Strukturen [Chr09, Mar06]. Allgemein wird zwischen den Optimierungstypen Dimensionierung (Sizing), Formoptimierung und Topologieoptimierung unterschieden. Die Dimensionierung ändert Dicken, Winkel und Längen. in einer Konstruktion. Die Formoptimierung verändert die Konturen eines Körpers. Sie bauen auf nicht unerhebliche Vorleistungen auf. Vorleistung bedeutet, dass ein rechenbares Modell der Struktur vorliegt. Die Topologieoptimierung dagegen verfolgt ein anderes Konzept [Roo04, Smi97, Che07, Ben95]. Sie versucht aus einem Block ein angepasstes Gerüst herauszuarbeiten indem sie bei jedem Iterationsschritt wenig belastetes Material eliminiert. Dabei ist der Block auch als rechenbares Modell anzusehen, aber er ist bei weitem nicht so spezifiziert, wie das fertige Modell der Formoptimierung und Dimensionierung. In diesem Kapitel wird auch noch die Funktionsweise der Topologieoptimierung kurz beschrieben werden, aber sie wird sonst nicht das Thema dieser Arbeit sein. In dieser Arbeit wird eine Formoptimierung bzw. Dimensionierung entwickelt werden.

Bei der Optimierung geht es darum zu verbessern. Ein bestehendes Modell soll also noch besser gemacht werden. Dabei geht man davon aus, dass ein mathematisches Modell für das zu optimierende Problem vorliegt. In diesem können dann die Eingabegrößen, die Variablen, geändert werden, um so die Ausgabegröße zu optimieren. Dies kann prinzipiell auf zwei Arten erfolgen. Zum einen führt man eine analytische Berechnung, eine Extremwertrechnung, mit der Differentialrechnung durch. Dann gibt es die zweite prinzipielle Möglichkeit, die algorithmische Optimierung, bei der man sich iterativ dem Optimum annähert. Dies ist die Möglichkeit, die für die Arbeit mit dem Computer prädestiniert ist, da bei ihr viele einzelne kleine Schritte und Wiederholungen den einen Schritt der analytischen Rechnung ersetzen und die Lösung näherungsweise erreichen. Die iterative Annäherung wird das Thema dieser Arbeit sein. Trotzdem werden zum Vergleich auch analytische Lösungen kurz beschrieben werden. Dazu werden die

theoretischen Hintergründe der iterativen Optimierung, die die Grundlage der in dieser Arbeit entwickelten Optimierung bilden, erläutert.

1.2 Analytische Berechnung des Optimierungsproblems mit einer Differentialrechnung

Ein Beispiel einer analytischen Rechnung mit einer Variablen ist das manchen schon aus der Schule bekannte folgende. Aus Blech soll eine Dose mit einem Inhalt von V_0 gefertigt werden. Die Frage der Optimierung sei, wie das Verhältnis von Radius zu Höhe zu wählen ist, damit möglichst wenig Material verbraucht wird.

Erst werden die Oberfläche der Dose und ihr Volumen als Formeln dargestellt. Auflösen nach h und Einsetzen in die Formel für die Oberfläche, liefert die Zielfunktion. Dies ist die Oberfläche in Abhängigkeit von dem Radius, also ein Optimierungsproblem mit einer Variablen. Die Minimierung der Zielfunktion erfolgt durch zweifaches Ableiten nach dem einzigen freien Parameter, nämlich r.

$$\textit{Oberfläche } A_D = 2\pi rh + 2\pi r^2$$

$$\textit{Volumen } V_D = \pi r^2 h = V_0$$

$$A_D(r) = 2\pi r \cdot \frac{V_0}{\pi r^2} + 2\pi r^2 = \frac{2V_0}{r} + 2\pi r^2 \;\rightarrow\; \textit{Zielfunktion}$$

$$A_D'(r) = -\frac{2V_0}{r^2} + 4\pi r = 0 \;\rightarrow\; r_{opt} = \sqrt[3]{\frac{V_0}{2\pi}}$$

$$A_D''(r = r_{opt}) = 12\pi > 0 \;\rightarrow\; \textit{Minimum}$$

Auch bei mehreren Variablen, f(x,y,..,n), lässt sich eine Extremwertberechnung durchführen. Bei mehreren Variablen arbeitet man mit den ersten und zweiten partiellen Ableitungen und der Hesse-Matrix. Dort berechnet man die stationären Punkte. An diesen Punkten sind alle ersten partiellen Ableitungen gleich null. Mit der Hesse-Matrix kann überprüft werden, ob die stationären Punkte ein Maximum oder Minimum sind. Weitere Stellen eines mehrdimensionalen Problems, die auf Extrema untersucht werden müssen, sind die Randpunkte, die Grenzen des angegebenen Definitionsbereichs. Die Berechnung der Randextremwerte bedarf größerer mathematischer Erfahrung und wird z. B. in [Mer95] beschrieben.

Bei dem gesonderten Fall, dass die Zielfunktion und die Randbedingungen linear sind, spricht man von der *Linearen Optimierung*. Dies ist deshalb ein Sonderfall, weil die Ableitungen wegen der Linearität verschwinden. Auch bei der Behandlung dieser Aufgaben sei auf die Fachliteratur verwiesen [Bom93].

Restringierte Probleme in der Strukturmechanik lassen sich z.B. mit der *Lagrange-Funktion* lösen[Bai94, Scu05, Shi09].

$$L(\vec{x};\vec{\lambda}) = f(\vec{x}) + \sum_{j=1}^{mg} \lambda_j \left(g_j(\vec{x}) + \mu_j^2 \right) + \sum_{k=1}^{mh} \lambda_k h_k(\vec{x}) \qquad \text{Gl. 1.1}$$

$f(\vec{x})$ ist die Zielfunktion, $g_j(\vec{x})$ sind Ungleichheitsrestriktionen, $h_k(\vec{x})$ sind Gleichheitsrestriktionen. μ_j^2 sind die Schlupfvariablen der Lagrangsche-Funktion.

Zur Berechnung des Optimums werden die partiellen Ableitungen von $L(x_i; \lambda_i)$ gebildet, gleich null gesetzt. Das entstandene Gleichungssystem wird gelöst. Restriktionen sind Nebenbedingungen. Es gibt Ungleichheitsrestriktionen, wie zum Beispiel eine Spannungsrestriktion, bei der eine zulässige Spannung nicht überschritten werden darf und es gibt Gleichheitsrestriktionen, die einer Variablen einen festen Wert zuordnen.

1.3 Algorithmische Optimierungsverfahren

Die oben gezeigten Beispiele lassen erahnen, dass der analytischen Rechnung Grenzen gesetzt sind [Spe93]. Sehr häufig ist die Aufgabe nur sehr schwierig und fehleranfällig analytisch zu lösen. Weiterhin kann es sein, dass die Zielfunktion nicht analytisch vorliegt. Dass wird in dieser Arbeit noch weiter geschildert werden, da ihre Ausführung und ihre Problemstellung genau diesen Fall betrifft. Dies ist zum Beispiel der Fall, wenn die Aufgabe zu komplex und umfassend ist. Mit Aufgabe ist nicht der Optimierungsablauf gemeint, sondern die Berechnung der Mechanik der Struktur. So ein Fall kann z.B. mit der FEM gelöst werden, da diese Methode genau für solche Fälle entwickelt wurde, in denen die Aufgabe für eine analytische Lösung zu komplex geworden ist. Aber die Folge für die Optimierung ist, wie schon oben erwähnt, dass keine Zielfunktion vorliegt, die analytisch gelöst werden kann.

Rechnergestützte Optimierungsverfahren sind einzusetzen wenn

- die Aufgabe nur sehr schwierig analytisch zu lösen ist:
- die Funktion f(x) nicht analytisch vorliegt

Der gebräuchlichste Spezialfall der rechnergestützten Berechnung ist die algorithmische Optimierung. Algorithmische Optimierungsverfahren nähern sich iterativ dem Optimum [Cha08].

1. Festlegung eines Startwerts x^k mit k=0
2. Änderung der Werte nach festgelegten Vorgaben
   $x^{k+1} = x^k + \Delta x^k$
3. Überprüfung der Abbruchkriterien

Wenn nicht erfüllt gehe zu 2

4. Optimale Lösung $x^* = x^{k+1}$

Die Optimierungstheorie steckt im Punkt 2 des Algorithmus. In diesem muss definiert sein, nach welchen festgelegten Vorgaben die variablen Werte geändert werden sollen. Diese Vorgaben, also die Optimierungsverfahren, sollen im Folgenden näher erläutert werden. Dazu sei an dieser Stelle gesagt, dass nur der prinzipielle Aufbau der Verfahren aufgezeigt wird und von den Verfahren auch nur die gängigsten behandelt werden. Denn es gibt viel zu viele Algorithmen, um hier auch nur ansatzweise alle zu nennen [Scwe77].

1.4 Das Beispiel 3 Stäbe

Folgend wird ein typisches Strukturoptimierungsproblem beschrieben (**Abbildung 1.4.1**). Dies ist als Hilfe gedacht, die Arbeitsweise der beschriebenen Optimierungsalgorithmen zu veranschaulichen. Es ist ein Modell aus drei Stäben, das in der angegebenen Weise mit der Kraft F belastet ist. Sein Volumen soll minimiert werden. Das Volumen ist damit die Zielfunktion. Bei den weiter unten beschriebenen Optimierungsverfahren gelten die Verfahren ohne Restriktionen. Die gesamte Aufgabe soll aber unter Beachtung der Ungleicheitsrestriktionen g_i optimiert werden. Die Ungleichheitsrestriktionen sind: Die Spannung in den Stäben muss kleiner sein als die zulässige Spannung. Dann bleiben die Verfahren mit Restriktionen anwendbar.

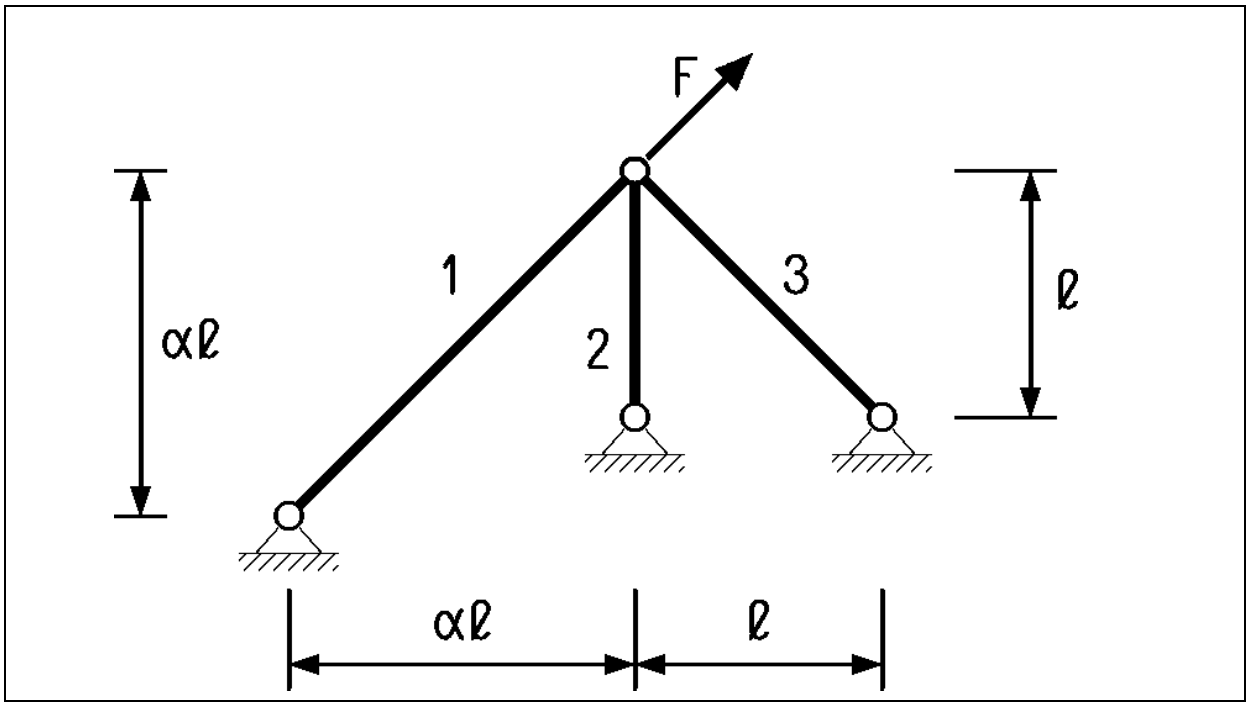


Abb. 1.4.1: 3 Stäbe

Gewichtsoptimierung bei Spannungsungleichheitsrestriktion

Zielfunktion: $V_S = (l_1 A_1 + l_2 A_2 + l_3 A_3)\rho$

l_i = Länge des Stabes i

A_i = Querschnittsfläche des Stabe i

ρ = spezifisches Gewicht

Restriktionen:

$$\sigma_1 = F \frac{\sqrt{2}A_3 + A_2}{\alpha A_2 A_3 + A_1\left(\sqrt{2}A_3 + A_2\right)}$$

$$\sigma_2 = F \frac{\alpha\sqrt{2}A_3}{\alpha A_2 A_3 + A_1\left(\sqrt{2}A_3 + A_2\right)}$$

$$\sigma_3 = F \frac{-\alpha A_2}{\alpha A_2 A_3 + A_1\left(\sqrt{2}A_3 + A_2\right)}$$

$$g_1 = 1 - \frac{\sigma_1}{\sigma_{zul}} \le 0$$

$$g_2 = 1 - \frac{\sigma_2}{\sigma_{zul}} \ge 0$$

$$g_3 = 1 - \frac{\sigma_3}{\sigma_{zul}} \ge 0$$

Zur Lösung dieses speziellen Problems wurde in Anlehnung an eine unveröffentlichte Arbeit von Herrn Professor Besdo ein Programm in Fortran geschrieben, das für einen festgelegten Bereich sämtliche Kombinationsmöglichkeiten ermittelt, zu diesen die Restriktionen überprüft und die beste zulässige Möglichkeit als Lösung bestimmt. Werte, die das Programm als Funktion von α liefert, sind:

Ein Maß für das „Gewicht“: $G = \alpha\sqrt{2}x + y + z$,

wobei die Querschnitte A_i auf $A_0 = \frac{F}{\sigma_{zul}}$ bezogen werden und damit:

$$x = \frac{A_1}{A_0}, \quad y = \frac{A_2}{A_0}, \quad z = \sqrt{2}\frac{A_3}{A_0}$$

das Gewicht des allein tragenden linken Stabes: g_0 $g_0 \equiv G_0 = \alpha\sqrt{2}$

die Spannungen $\frac{\sigma_i}{\sigma_{zul}}$

Es wurden Werte für α in der Umgebung des Wertes 2 untersucht, da dort die Lösung der Optimierung „umschlägt“. Bei α<2 übernimmt Stab eins die größte Last und die Stäbe 2 und 3 werden sehr dünn. Bei α>2 ist es genau andersherum, die Stäbe 2 und 3 übernehmen die Last und Stab 1 wird sehr dünn.

In **Abbildung 1.4.2** sind einige der Berechnungen für unterschiedliche Alphawerte dargestellt.

Alpha	x	y	z	g_0	G_0
1	1	1E-7	1E-7	1,4142135	1,4142139
1,5	1	1E-7	1E-7	2,1213203	2,1213207
1,99	1	1E-7	1E-7	2,814284	2,8142854
2,001	4,4E-2	1,35	1,35	2,82856	2,82845
2,2	1E-7	1,416	1,416	3,11	2,832

Abb. 1.4.2: Ergebnisse

Auffällig ist bei α=2,2 der Wert g_0=3,11, der besagt, dass der Stab 1 alleine den Volumengleichwert 3,11 hätte, während die Stäbe 2 und 3 gemeinsam den Volumengleichwert 2,832 besitzen. Dies erklärt das Ergebnis der Optimierungsrechnung, die das Volumen von Stab 1 gegen Null laufen lässt.

Nach der Einführung der für diese Arbeit entwickelten Algorithmen in Kapitel drei soll mit ihnen die hier ermittelte Ergebnisentwicklung bestätigt werden.

1.5 Optimierungsaufgaben ohne Restriktionen

Bei dem Fall einer Aufgabe ohne Restriktion zählt nur, wie schon weiter oben angedeutet wurde, die Zielfunktion. Restriktionen g_i werden nicht in Betracht gezogen. Die Minimierung des Gewichts ohne Restriktionen bedeutet bei Strukturoptimierungen, dass das Minimum am Rand des Definitionsbereiches liegt, also an den unteren angegebenen Grenzen der Durchmesser der Stäbe. Dies ist kein überraschendes Ergebnis, sondern soll nur zur Veranschaulichung der Algorithmen dienen. Ein „richtiges“ Optimierungsproblem ist die Beispielaufgabe des Abschnitts 1.4 erst mit ihren Restriktionen.

Gängige Optimierungsverfahren sind die *ableitungsfreien Suchverfahren*, die so genannten Verfahren nullter Ordnung. Sie funktionieren durch wiederholte Bestimmung des Zielwertes $V(a_i)$ durch Variation der Variablen a_i. Im Falle der Beispielaufgabe 3 Stäbe bedeutet das, dass die Dicken der Stäbe je nach verwendetem Algorithmus, gleichzeitig oder nacheinander mit unterschiedlichen Schrittweiten geändert werden, dann die Aufgabe gelöst wird, also das Volumen berechnet und dann das Ergebnis des Zielfunktionswertes mit alten Zielfunktionswerten oder mit einem vorgegebenen Zielfunktionswert verglichen wird. Abbruchbedingungen können eine vorgegebene Differenz zu einem vorgegebenen Wert sein, oder auch nicht mehr eintretende Verbesserungen des Zielfunktionswertes. Dies ist eine allgemeine Bedingung und ist nicht nur für die ableitungsfreien Suchverfahren relevant. Diese Suchverfahren gelten als langsam

aber sehr robust und können auch eingesetzt werden, wenn die Zielfunktion nicht differenzierbar ist oder schwach unstetig.

Weiterhin gibt es die *Intervallverfahren*. Intervallverfahren teilen das untersuchte Intervall in Bereiche auf. Die Bereiche, in denen das Minimum wahrscheinlich nicht liegt, werden aus der Untersuchung herausgenommen.

Einen anderen Ansatz stellen die *Evolutionsstrategien* dar. Zuerst werden die Eltern, die Startvektoren erzeugt. Mit Hilfe eines Zufallsgenerators werden die neuen Vektoren, die Nachkommen erzeugt. Diese liegen normalverteilt in der Umgebung der Eltern. Die besten Nachkommen werden zur weiteren Untersuchung des Falls herangezogen, die schlechteren werden verworfen. Die Schrittweite wird über das Verhältnis von verbesserten Zielfunktionswerten zur Gesamtzahl der Nachkommen gesteuert [Zhu08, Laum02].

Ein sehr verbreiteter Ansatz sind die *Gradientenverfahren*. Sie benutzen die Ableitungen und sind unterteilt in die Verfahren erster und zweiter Ordnung, wenn sie mit der ersten oder entsprechend der zweiten Ableitung arbeiten. Die Verfahren erster Ordnung ermitteln mit den Ableitungen die Gradienten des Feldes der Zielfunktion. Der negative Gradient ist der „steilste Abstieg“, also der vermutlich schnellste Weg zum Optimum. Es gibt 2 Möglichkeiten die Gradienten zu ermitteln. Zum einen gibt es das *Differenzenverfahren*. Dabei werden die Variablen variiert, und so die Änderungen (Gradienten) ermittelt. Zum anderen können die Gradienten, wie in dem Fall der Beispielaufgabe, bei dem die Gleichungen vorliegen, auch durch partielles Ableiten in einem Punkt berechnet werden. Dies gilt auch für die Aufgaben mit Restriktionen. Der Ablauf der Optimierung ist dabei prinzipiell der Gleiche. Die Richtung des Schrittes wird durch die Gradienten ermittelt. Dann muss eine Schrittweite festgelegt werden. Danach muss das mechanische Modell berechnet werden, um den Wert der Zielfunktion zu ermitteln. Es folgt der Abgleich mit dem Abbruchkriterium und, wenn es nötig ist, ein neuer Iterationsschritt [Esc06, Pad04, Pha07].

Das Verfahren der *konjugierten Gradienten* arbeitet im Prinzip genauso wie die anderen Gradientenverfahren. Der Unterschied ist, das konjugierte Gradienten die Informationen vorangegangener Informationsschritte zur Bestimmung von Suchrichtung und Schrittweite nutzen. Wie bei allen Weiterentwicklungen von Verfahren hat man dadurch Vor- und Nachteile. Die Effizienz kann sich steigern, der Algorithmus kann sich aber auch verschlechtern und langsamer werden, in eine „falsche“ Richtung laufen oder abbrechen.

Das gleiche gilt für die Gradientenverfahren zweiter Ordnung. Diese nutzen die zweite Ableitung und die Hesse-Matrix, um die Konvergenz zu erhöhen. Jedoch kann

sich durch die größere Komplexität der Rechenaufwand erhöhen oder es kann numerische Probleme geben.

Hier sei angemerkt, dass die hier beschriebenen Verfahren in ihrer Darstellung keine eindeutig definierten Algorithmen sind sondern prinzipielle Darstellungen, aus denen man Algorithmen erstellen kann. Das bedeutet auch, dass es nicht nur ein Gradientenverfahren gibt. Es gibt theoretisch unendlich viele Ausprägungen, je nachdem wie das Verfahren umgesetzt wurde. Dies hat zur Folge, dass sehr viele Algorithmen schon entwickelt wurden, so dass es unmöglich ist, alle aufzuzählen. So werden in diesem Kapitel die Prinzipien der Algorithmen aufgezeigt. Auch zu den Prinzipien ist zu sagen dass es sehr viel mehr gibt, als die hier genannten. Dies sind nur diejenigen, die in der Strukturoptimierung häufig gebraucht werden. Dies gilt natürlich auch für die Algorithmen der Optimierungsaufgaben mit Restriktionen.

1.6 Optimierungsaufgaben mit Restriktionen

Die allgemeine Form der Optimierungsaufgabe ist eine Aufgabe mit Restriktionen. Restriktionen sind Nebenbedingungen, die einzuhalten sind. Der klassischste Fall für eine Optimierungsaufgabe in der Strukturmechanik ist das oben beschriebene Beispiel der Gewichtsminimierung bei Beachtung der Nebenbedingung der zulässigen Spannung. Erst mit der Nebenbedingung wird die Beispielaufgabe 3 Stäbe interessant, da nun das Ergebnis völlig unbekannt ist. Unbekannt war das Ergebnis bei der unrestringierten Aufgabe zwar auch, aber man wusste, dass das Optimum bei der jeweils angegebenen Untergrenze der Variablen lag. Nun muss das Optimum gefunden werden, das die Nebenbedingung erfüllt.

Straffunktionsverfahren sind mit dem oben genannten Verfahren der Lagrange-Funktion verwandt, sind aber nicht dasselbe. Bei den Straffunktionsverfahren geht es darum, die Funktion anzunähern und im Falle der Verletzung der Restriktion, sehr hohe Aufschläge auf den Zielfunktionswert zu generieren, ansonsten aber wie eine Aufgabe ohne Restriktion zu berechnen. So ergibt sich unter Einbeziehung der 3 Stäbe eine Minimierungsaufgabe in der unten dargestellten Form.

$$Min\left\{ z_r = z(A_1, A_2, A_3) + r * \frac{1}{(g_1 + g_2 + g_3)} \right\} \qquad \text{Gl. 1.2}$$

Diese Gleichung wird mit den Methoden der unrestringierten Probleme schrittweise optimiert. Die passende Bestimmung des Faktors r ist sehr wichtig. Die Größe von r muss dem Problem angepasst werden. Für diese Art der Optimierungsrechnung sind die Gleichungen des Problems notwendig. Mit einem reinen Differenzenverfahren

ohne Kenntnis der die Restriktion beschreibenden Gleichungen wäre diese Aufgabe so nicht zu lösen.

Die *Direkten Verfahren* arbeiten ohne Straffunktionen. Eines dieser Verfahren ist die Methode der *zulässigen Richtungen*. Die Methoden der zulässigen Richtung arbeiten mit der Schnittmenge von Bereichen. Diese Schnittmenge bildet sich aus dem Bereich der verbesserten Werte und dem zulässigen Bereich. Dies bedeutet, sie brauchen Informationen über die Restriktionsgrenze, um mit ihrer Hilfe und dem Gradienten verbesserte und zulässige Werte zu erhalten.

Die *Approximationsverfahren* versuchen die Effizienz zu erhöhen. Die größte Rechenzeit bei der Optimierung wird durch die Sensitivitätsanalyse und die FEM-Analyse verursacht. Um diese zeitintensiven Anteile zu minimieren gibt es die Approximationsverfahren. Zu Beginn eines Optimierungsschritts werden die Systemgleichungen durch Polynome approximiert. Dadurch soll die Effizienz gesteigert werden. Eine einfache Form der Approximation ist die Sequentielle Lineare Programmierung. Zielfunktionen und Restriktionen werden an einer Stelle in Taylor-Reihen entwickelt und hinter dem linearen Glied abgebrochen. Wenn ein Gradientenverfahren genutzt wird, müssen pro Optimierungsschritt die Gradienten nur einmal berechnet werden. Diese linearisierte Optimierungsaufgabe kann nun relativ effizient mit dem *Simplex-Verfahren* gelöst werden. Der Simplex-Algorithmus ist ein Verfahren der linearen Programmierung, also des Lösens linearer Optimierungsprobleme und soll hier nicht weiter erläutert werden [26]. Bei der quadratischen Approximation bricht man entsprechend die Taylor-Reihe nach dem quadratischen Glied ab [Ble03, Häu03, Lau03].

Für den Fall, dass die Gradienten nicht analytisch bestimmt werden können, oder die Systemantworten hochgradig nichtlinear sind und stark oszillieren gibt es die *Succesive Response Surface Method*. Um den aktuellen Punkt wird bei jeder Iteration die Approximation geglättet. Dazu wird ein Satz von Stützstellen durch die Approximation gelegt.

Die oben genannten Verfahren sind die Grundlage für die meisten in der Strukturoptimierung genutzten oder der für sie entwickelten Algorithmen.

1.7 Optimierung auf der Basis der Finite-Elemente-Methode (FEM)

Die Berechnung von Struktur- und mechanischen Modellen wird mittlerweile zum allergrößten Teil mit FEM durchgeführt, da mit ihr komplexe Probleme gelöst werden können, die analytisch oder mit anderen Näherungsverfahren als der FEM nicht mehr durchzuführen sind. Aus diesem Grund liegt es nahe, die FEM in die

Strukturoptimierung einzubeziehen [Ped08]. Dazu gibt es mehrere Ansätze. Zum Beispiel gibt es interne und externe Optimierer.

1.8 Interne Optimierer

Interne Optimierer sind in den FEM Solver integriert. Sie sind jedoch in ihrer Anwendung sehr starr und sehr begrenzt. Beispielsweise beschränkt sich in dem FEM-Programm *Abaqus* das interne Optimieren auf die Sensitivitätsanalyse. Diese Analyse wird semianalytisch genannt und rechnet mit der FEM-internen Gleichung

$$\overline{\overline{K}}(x) * \vec{u} = \vec{F} \qquad \text{Gl. 1.3}$$

Die Gleichung wird partiell abgeleitet

$$\frac{\partial \overline{\overline{K}}}{\partial x_i}\vec{u} + \overline{\overline{K}}\frac{\partial \vec{u}}{\partial x_i} = \frac{\partial \vec{F}}{\partial x_i} \qquad \text{Gl. 1.4}$$

Hierbei wird für jede Variable ein Gleichungssystem erstellt, das gelöst werden muss. D. h., es wird eine Rechnung, ein so genannter Pseudolastfall; durchgeführt. Aber es muss nicht jedes Mal die Steifigkeitsmatrix K neu erstellt und faktorisiert werden. Dies spart Rechenzeit.

Bei sehr großen Variablenzahlen wäre die Bestimmung jeder partiellen Ableitung, also das Lösen des Pseudolastfalls sehr aufwendig. Für so einen Fall wurde die adjungierte Methode entwickelt. Bei ihr wird das totale Differential mathematisch so umgestellt, dass nicht für jede Variable ein Pseudolastfall gelöst werden muss [Aba10, Bai94, Har08].

1.9 Externe Optimierer

Externe Optimierer sind die Programme, die nicht in den jeweiligen Solver integriert sind. Dies sind zu einem sehr großen Teil die Topologieoptimierer. Topologieoptimierung dient der Vorauslegung eines Bauteiles. Von der erhaltenen Struktur aus müssen noch weitere Berechnungen durchgeführt werden, oder folgende Optimierungsrechnungen mit einem Parameteroptimierungsprogramm. Die Topologieoptimierung ist eine andere Vorgehensweise als die in dieser Arbeit vorgestellte. Ihre prinzipielle Funktionsweise soll hier aber kurz erläutert werden. Bei der Topologieoptimierung können die Variablen die E-Moduln und die Dichte der einzelnen Elemente sein. Im Laufe der Optimierung erhält man unterschiedliche Dichteverteilungen. Die Proportionalität des Elastizitätzmoduls ist in der vierten Potenz des Dichtefaktors, so dass die Elastizität mit der Dichte sehr schnell abnimmt,

was zur Folge hat, dass meist die Extremalzustände angestrebt werden. Das bedeutet, dass man entweder den Wert eins für die Dichte erhält oder den minimalen Wert nahe null und nach Möglichkeit nicht die Werte dazwischen. Die Elemente mit dem Wert eins ergeben die neue Form der Struktur [Che07, Coe08, And05].

Ein Programm, das dieses Verfahren einsetzt, ist das Topologieoptimierungs-programm *Tosca* von *FE-Design* [Grü03, Sau09,Lau08, Köt03, Bog03, And03, Puc03]. Tosca ist plattformunabhängig und zum Beispiel zur Vernetzung mit dem FEM-Programm *Ansys* in der Lage, so dass die Modellerstellung in Ansys stattfindet und zur Berechnung dessen Solver genutzt wird, während die Steuerung der Optimierung von Tosca durchgeführt wird. Ein ähnliches Programm ist *Opti Struct* von *Altair* [Har06].

Für die Form- und die Dickenoptimierung mit der FEM werden meistens universelle Optimierer eingesetzt. Dies sind Optimierungsprogramme, die primär nichts mit FEM-Programmen zu tun haben, jedoch mit ihnen in Verbindung gebracht werden. Dies sind zum Beispiel das Programm *OptiSlang* von *Dynardo* [Lut09, Kel09, Wil06, Wil08, Sem06]. Dieser Optimierer beinhaltet Gradientenverfahren, evolutionäre Algorithmen und Sequentielle Lineare Programmierung. Weiterhin gibt es *Heeds* [Sid04], das ein Simplexverfahren, evolutionäre Algorithmen, *simulated annealing*, *response Surface* und *quadratic programming* durchführen kann. Eine weitere Algorithmensammlung ist das Programm *ISight* [Pad04], das über die oben in der Einleitung beschriebenen Verfahren verfügt.

Es gibt einige Arbeiten, die das Thema haben, mit FEM und diesen oben genannten Algorithmensammlungen Optimierungen durchzuführen. Dies sind allerdings keine universellen vollständigen Optimierungsprogramme. In diesen Arbeiten sind die Programme zusammengeführt worden, um damit einzelne Probleme zu bearbeiten. So wurden mit Abaqus/CAE und Heeds eine Gummimuffe und ein Metallrahmen optimiert [Sid04]. Bei dem Rahmen wurde bei einer Dickenoptimierung die Steifigkeit ohne Gewichtszunahme um 12 % erhöht. Genauso wurde ein Versuch gestaltet, eine Lüfterabdeckhaube mit Ansys und OptiSlang zu optimieren. Hierbei stand der Versuch im Vordergrund, die Optimierungsschleife zu automatisieren. Die Optimierung sollte CAD-orientiert arbeiten, das heißt, die neuen Variablen sollen die Geometrie im CAD-File ändern. Erst stand der Versuch, auf *Pro-Engineer* zuzugreifen. Dies war jedoch nicht möglich. Die Ansys-eigene Oberfläche *Design Modeller* ließ sich jedoch in die Optimierungsschleife integrieren. Den Optimierungsalgorithmus stellt OptiSlang zur Verfügung. Dies ist ein evolutionärer Algorithmus [Ege07].

Eine universellere Form der Optimierung ist die Templatetechnik von *Noesis* mit ihrem Programm *Optimus*. Mit diesem ist die Optimierungsschleife geschlossen [Qua05]. Modellerstellung und FEM-Berechnung geschehen in einem FEM-

Programm. Templatetechnik bedeutet dabei, dass auf die Eingabefiles der Solver zugegriffen wird. Das heißt, es können Dicken von Schalenelementen und Balkenelementen verändert werden. Der Veränderung der Geometrie und des Netzes sind aber Grenzen gesetzt. So ist es prinzipiell nur möglich einzelne Knoten zu verschieben, was durch die damit entstehende sehr große Variablenzahl die engen Grenzen setzt [Sce10, Zha08, Nit05]. Die gleichen Aussagen gelten für das Templateprogramm *CAOSS* [Mes00].

Nach [Qua05] liegt die Problematik der automatischen Strukturoptimierung nicht nur allein in der Nichtlinearität dieser Aufgaben, sondern auch in der Gestaltung des Optimimierungsprozesses. Die Forderung nach Stabilität stellt hohe Qualitätsansprüche an die Softwareprodukte und vor allem an die Schnittstellen-kommunikation. Das Zusammenspiel von *SFE Concept*, *LS-OPT*, *Optimus* und *LS-DYNA* führte in der Untersuchung zu keinem akzeptablen Ergebnis [Scä08, Bös06, Hör04, Hop04].

1.10 Ziel dieser Arbeit

In dieser Arbeit geht es um die Erstellung solch einer universellen Parameter-optimierung. Diese Parameteroptimierung soll dem Konstrukteur und dem Berechner ein Hilfsmittel sein, um eine bestimmte Konstruktion hinsichtlich der Annäherung an eine bestimmte Anforderung zu erreichen. In diesem Zusammenhang wird eine Konstruktion durch eine Reihe von festen (unveränderten) Daten und veränderbaren Daten, den Parametern beschrieben. Die Parameteroptimierung soll dann bestimmte Parameter des Modells so lange verändern, bis eine bestimmte Qualität, z. B. das Unterschreiten eines kritischen Spannungsmaßes, hinreichend gut erreicht ist. Dies wird durch die Berechnung eines Qualitätsmaßes unter Betriebsbelastung festgestellt. Werkzeug zur Berechnung des Qualitätsmaßes ist ein FEM-System. Die Veränderung der Konstruktion durch Manipulation der Konstruktionsparameter soll durch ein weitgehend automatisiertes algorithmisch gesteuertes Programmsystem (Programmskript) geschehen. Hier wird die Systematik des Programms ABAQUS/CAE und seiner Skriptsprache *PYTHON* sowohl als zentrales Datenhaltungssystem als auch als Berechnungstool verwendet.

Der Ablauf der Programmierung ist in einem Python-Skript formuliert. In diesem Skript wird die Optimierungsstrategie definiert. Mit ihr wird für jedes Optimierungsproblem festgelegt, mit welchen Algorithmen in welcher Reihenfolge die Optimierung durchgeführt wird. Durch die Erstellung dieser Rechenvorschrift soll der Computer dazu genutzt werden, optimale, zumindest aber optimierte Lösungen eines Problems zu ermitteln, womit sonst der Konstrukteur in einem langwierigen iterativen Konstruktionsprozess beschäftigt wäre.

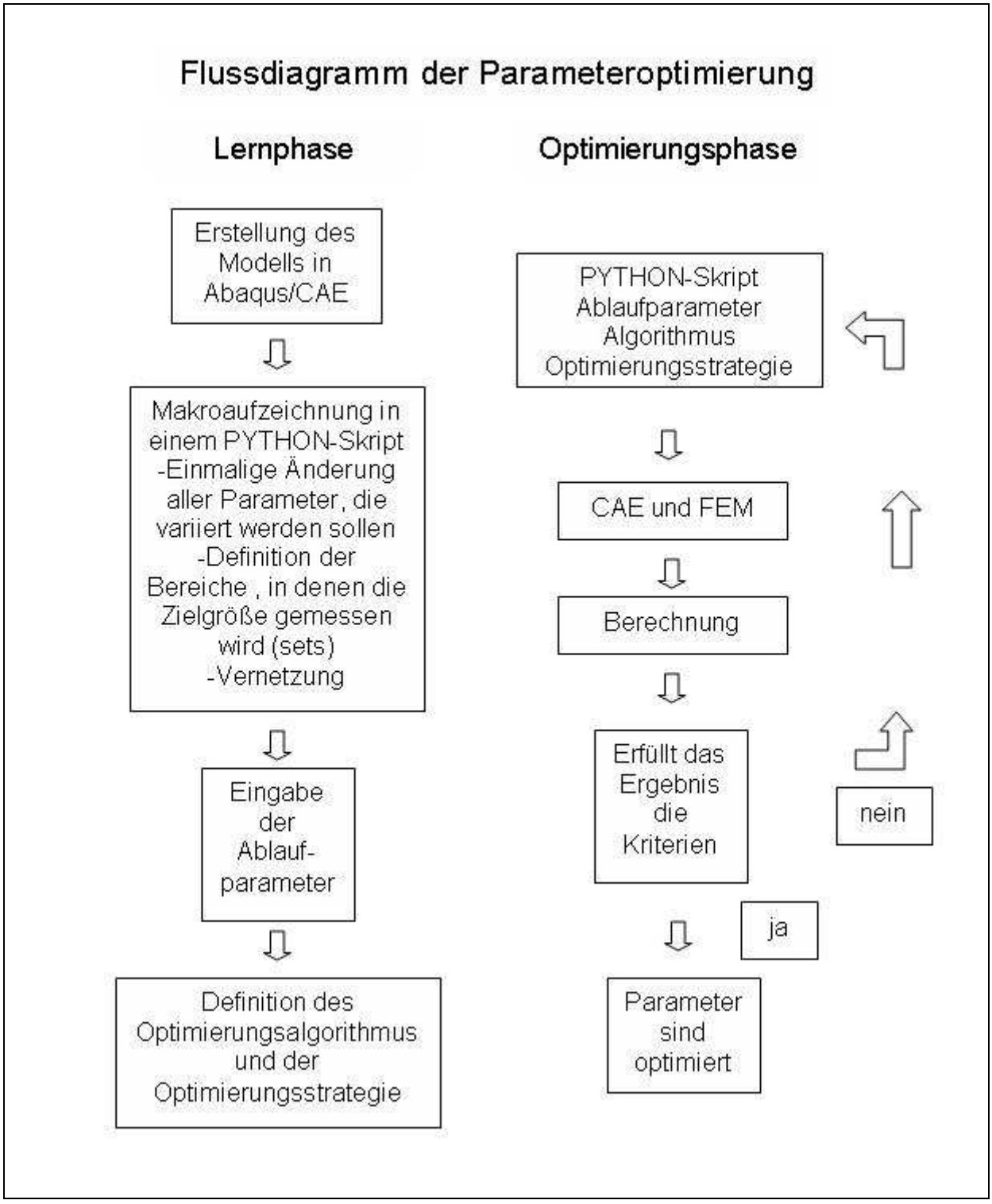


Abb. 1.10.1: Ablauf der Optimierung

Der Ablauf einer Optimierung ist wie folgt vorgesehen **(Abbildung 1.10.1)**: Die gesamte Optimierung ist in die Lernphase und die Optimierungsphase unterteilt. In der Lernphase wird der prinzipielle Ablauf der Optimierung festgelegt. Ein FEM Modell, in dem die zu variierenden Parameter ausgewählt werden, wird erstellt. Dazu wird eine Makroaufzeichnung gestartet, die alle Aktionen aufzeichnet. In dieser

Aufzeichnung werden alle zu variierenden Maße (=Parameter) gekennzeichnet. Diese Parameter können geometrisch oder werkstofflich sein. Danach werden die Bereiche des Bauteils, die für die Untersuchung in Betracht kommen (Faces, Cells) vernetzt.

In einer Schleifensteuerung müssen methodenbezogene Ablaufparameter für die Optimierungsschleife, wie z.B. die Startschrittweite, die maximale Schrittweite und die Zieltoleranz, definiert werden. Des Weiteren muss der zu verwendende Optimierungsalgorithmus als Code-Datei vorliegen. Mit diesen Mitteln kann die Optimierung, bzw. die Optimierungsphase gestartet werden. Dies geschieht durch das Aufrufen des Skriptes, welches die Berechnung nach jedem Optimierungs- und Berechnungsschritt mit systematisch veränderten Parametern immer wieder neu startet, bis die erwartete Zielgröße (z.B. Spannungen oder Dehnungen) oder ein anderes Abbruchkriterium erreicht ist.

Hier seien einige Anforderungen an die Parameter und die Optimierungsstrategie genannt. Unter Parametern sollen skalare Größen verstanden werden, die geometrische Maßeinheiten, Materialkenngrößen u.v.A.m. sein können. Sie können durch Rand- und Nebenbedingungen eingeschränkt oder miteinander verknüpft sein. Neben die Parameter, deren Werte während der Optimierung verändert werden, treten gleichartige „Fixe Daten“ auf, die während der Optimierung unverändert bleiben. Anzahl und Charakter von Parametern und fixen Daten müssen so gewählt sein, dass sie die Konstruktion eindeutig festlegen.

Wenn eine Konstruktion mit Betriebslasten beaufschlagt wird, so ergeben sich bestimmte messbare (oder errechenbare) Reaktionen (z.B. Materialbeanspruchungen, Verschiebungen, u.s.w.). Aus diesen Reaktionen wird der Wert der Gütefunktion, die die Tauglichkeit einer Konstruktion beschreibt, eines bestimmten Parametersatzes errechnet. Ein Parametersatz sind die Parameter, die während eines Optimierungsschrittes geändert werden. Ziel eines Optimierungsschrittes ist es, durch gezielte Veränderung eines Parametersatzes den Wert der Gütefunktion zu erhöhen. Die Optimierung soll beendet sein, wenn ein geeignetes Abbruchkriteium (z.B. Stationarität der Gütefunktion, Gütefunktion erreicht ein bestimmtes Maß, u.s.w.) erreicht bzw. überschritten wird. Die Entwicklung einer Konstruktion (eines Parametersatzes) entlang der einzelnen Optimierungsschritte lässt sich als gerichtete Evolution begreifen, bei der die Optimierungsrichtung und –schrittweite von Algorithmen gesteuert werden, die die Entwicklung der Gütefunktion verarbeiten. Es ist zu erwarten, dass es bei komplexen Problemen sinnvoll ist, je nach Status der Optimierung, verschiedene Algorithmen anzuwenden, deren Abfolge in einer Optimierungsstrategie festgelegt wird. Der Ablauf einer Optimierung ist das Wechselspiel von der Feststellung der Güte der Konstruktion (des Parametersatzes) durch Berechnung der Beanspruchung in Folge Betriebsbelastung auf der einen Seite und der gezielten Veränderung der Konstruktion durch Manipulation des

Parametersatzes auf der anderen Seite.

Zur prinzipiellen Darstellung der Algorithmen sei hier ein sehr einfacher Fall aufgezeigt. Ein senkrechter fest eingespannter Balken soll in seiner Dicke optimiert werden, so dass σ_{zul} nicht überschritten wird. In der Eingabemaske werden nun die Startschrittweite Δd, σ_{zul} und der Fehler f eingegeben. Der Algorithmus vergrößert nun so lange die Dicke des Balkens, bis σ_{zul} unterschritten wird. Dann verringert er die Dicke mit ständig abnehmender Schrittweite, bis sich die Spannung innerhalb des eingegebenen Fehlers befindet.

Zu überlegen wäre auch, ob eine Optimierung von Oberflächenformen an diskreten Punkten machbar ist. Diese so geformten Bauteile müssen in der Lage sein, Kräfte besser zu übertragen, bzw. Spannungen im Bauteil zu minimieren. Auch das Einbeziehen von Eigenfrequenzen als Nebenbedingung kann ein Optimierungskriterium sein. Auf jeden Fall sind die Entwicklungsmöglichkeiten für die Optimierungsstrategien immens. So kann man Strategien entwickeln, die entscheiden können, welche Parameter in Systemen, die aus mehreren Bauteilen bestehen, in welchen Grenzen, mit welcher Priorität und mit welchem Algorithmus optimiert werden. Weiterhin ist zu untersuchen, ob mathematische Hilfsmittel wie Interpolationsverfahren die Anzahl von Simulationsrechnungen verringern können.

Die in dem Programm genutzten Algorithmen werden nicht aus einer Algorithmenbibliothek stammen, sondern werden Teil der Entwicklung sein und so auf die behandelten Probleme und deren Typisierung spezialisiert sein.

Erklärtes Ziel dieser Arbeit ist die Steigerung der universellen Einsetzbarkeit. Es gibt keine Schnittstellen zu anderen Programmen, die die Robustheit beeinträchtigen könnten. In *CAE* erstellte Modelle sollen auch variiert werden können. Die Definition der Variablen wird in *CAE* geschehen können und die Eingabe der Optimierungsparameter geschieht in einer Programmierten Eingabemaske. Durch die Implementierung eines Parameterverfahrens, dessen Parameter in der Abaqus-internen grafischen Oberfläche *CAE* definiert und verändert werden, ist die Möglichkeit der strukturellen und geometrischen Änderung kaum begrenzt. Dadurch müssen auch kaum Annahmen und Voraussetzungen getroffen werden und erfüllt sein. Die originale Abbildung des Ausgangsproblems in den Optimierungsablauf ist also vollständig gegeben.

2. Aufbau der Optimierung

Die Optimierung gemäß dieser Arbeit ist ein Anwendungswerkzeug für Ingenieure, die Bauteile optimieren möchten. Dies geschieht auf Basis der FEM, um deren Vorzüge, wie die Möglichkeit komplexe Strukturen abzubilden und zu berechnen, direkt für die Optimierung nutzen zu können. Der Optimierung liegen zwei Hauptbereiche zugrunde. Ihr Kern sind ihre Algorithmen, die für den Ablauf des Rechengangs verantwortlich sind und vorgeben, wie der Weg durch das Feld der Zielfunktion sich gestaltet. Diese Algorithmen werden in Kapitel 3 behandelt werden. Wie dagegen die Optimierung prinzipiell aufgebaut ist und wie der Zugriff auf die Optimierungsumgebung aussieht, wird in diesem Kapitel geschildert. Weiteres zu diesem Bereich ist im Anhang A nachzuvollziehen.

Diese Optimierung ist eng an das kommerzielle Finite-Elemente-Programm Abaqus der Firma Simulia und deren graphische Oberfläche Abaqus/CAE gebunden. Mit ihr können die Modelle erstellt werden, die die Grundlage für die FEM-Rechnung sind. Zu der Modellerstellung gehören die Definition der Geometrie der Bauteile und der Materialeigenschaften, die Positionierung, Kontaktbedingungen, Belastungen, der Aufbau des FE-Netzes und die Festlegung des FEM-Codes. Diese gesamte Modellerstellung geschieht durch Zeichnen in der Oberfläche und durch das Ausfüllen der Pop-Ups bzw der Fenster der Oberfläche. Diese Modelle bilden eine Grundlage der Optimierung, denn auf diese Modelle muss die Optimierung automatisiert zugreifen können um sie zu verändern, was zur Verbesserung der Modelle nötig ist. Abaqus bietet die Möglichkeit dieses automatisierte Zugreifen programmtechnisch umsetzen zu können mit der Erweiterung *Graphical User Interface (GUI)*. Damit ist es möglich, die Grafische Oberfläche von CAE zu verändern und neue Reiter in Menüleisten und neue Fenster zu erzeugen. Mit diesen äußerlichen Veränderungen können neu erstellte Ausführungsbefehle und Funktionen verknüpft werden. Auf diese Weise ist die Optimierung als Anwendungswerkzeug erstellt worden.

CAE wurde mit der Programmiersprache *PYTHON* erstellt. Demzufolge wurde auch die Optimierung als GUI-Erweiterung in dieser Sprache geschrieben. Weiterhin gibt es ein Protokoll jeder CAE-Sitzung, das auch in PYTHON mitdokumentiert wird, das so genannte *Replay-File*. Diese Eigenschaft macht es zum wesentlichen Teil der Lernphase der Optimierung, da man das Replay-File als Skript auch „abspielen“ kann

und dabei die eingegebenen Befehle ausgeführt werden. Dies lässt sich in die Optimierung integrieren. Wird nun die Änderung der Variablen einmal vom Anwender durchgeführt, wird dies aufgezeichnet und wird in der Optimierung automatisch beliebige Male wiederholt.

Die Ergebnisse jeder einzelnen Optimierungsrechnung werden in die Ergebnisdatei, die Datei *ergebnis.odb* geschrieben. Die Auswertung der Ergebnisse und deren Visualisierung wird auch in Abaqus/CAE durchgeführt. Es ist aber auch möglich auf die Ergebnisdatei mit der Scriptsprache Python zuzugreifen und einzelne Werte zu extrahieren. Dies ist für die Durchführung der Optimierung von Belang.

2.1 Ablauf und Struktur der Optimierung

An dieser Stelle sei noch einmal auf die **Abbildung 1.10.1** verwiesen, in dem die Optimierung dargestellt ist. Dort ist zu sehen, dass der Prozess der Optimierung in die Lernphase und die Ausführungsphase unterteilt ist. In der Lernphase wird das zu optimierende Modell beispielhaft geändert und danach vernetzt. Diese Informationen werden in Skriptsprache im Replay-File gespeichert.

Die beispielhafte Änderung bedeutet, dass jede Variable einmal geändert wird. Der Wert der Variable ist in diesem Fall unerheblich, da er nichts mit den untersuchten Werten der Optimierung zu tun hat. Was lediglich zu beachten ist, ist dass die Werte zur Identifikation dienen müssen. Daher wurde die Festlegung getroffen, dass die Werte die Ziffernfolge *0001* enthalten müssen, z.B. 56,0001 mm als Wert für einen Durchmesser. In der weiteren Aufarbeitung des Replay-Files werden so die Variablen identifiziert. Auf diese Weise kann der Wert 56,0001 durch z.B. Variable1 ersetzt werden, da ja in dem Ablauf der Optimierung variable Werte angenommen werden müssen. Die erste identifizierte Ziffernfolge 0001 des Replay-Files erhält die Zuordnung zur Variable 1, die zweite wird die Variable 2 usw.. Weiterhin gehören zur Lernphase die Definition des Auswertungssets und die Vernetzung des Modells. In das Auswertungsset gehören die Elemente oder die Knoten, an denen die Ergebniswerte herausgeschrieben werden sollen.

Gestartet wird die Lernphase durch den Anwender des Programms, indem er den Reiter *Optimierung* in der modifizierten CAE-Oberfläche anwählt und dann *StartLern* auswählt (**Abbildung 2.1.1**). Dadurch wird lediglich eine Markierung in das Replayfile geschrieben. Nach Durchführung der Lernphase wird *EndLern* angeklickt, wodurch eine zweite Marke gesetzt wird. Zwischen diesen beiden Marken werden nun die Variablen identifiziert und aufbereitet und dieser Komplex (zwischen den beiden Marken) wird als VariationsModul.py abgespeichert.

Nach dem Speichern der Lernphase wird die Ausführungsphase der Optimierung festgelegt.

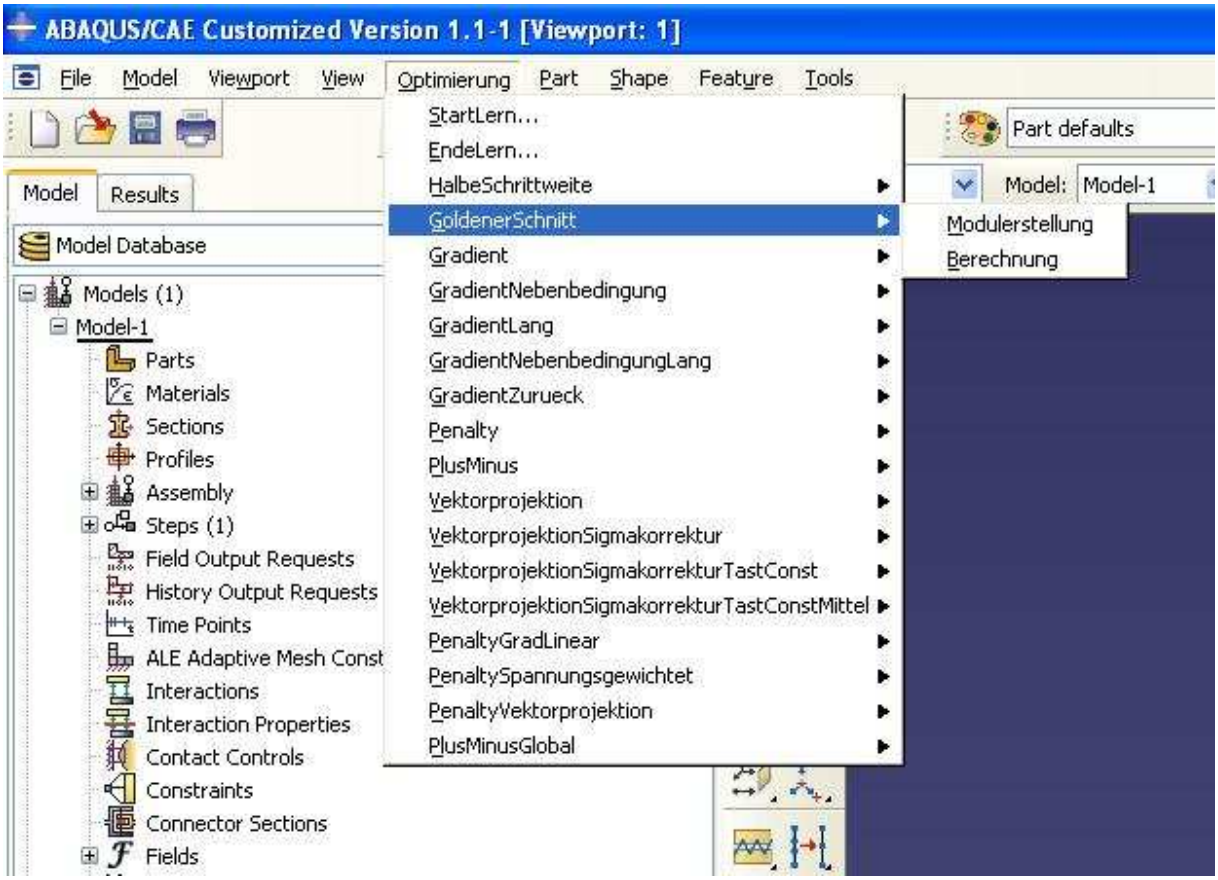


Abb. 2.1.1: Optimierungsmenü im Hauptfenster

In **Abbildung 2.1.1** sind sämtliche verfügbare Algorithmen ersichtlich. Ihre jeweilige Funktionsweise wird in dem folgenden Kapitel beschrieben werden. Um die Optimierung festzulegen, muss ein Algorithmus ausgewählt werden. Für die Definition des Algorithmus sind zwei Schritte notwendig. Zuerst muss die *Modulerstellung* ausgefüllt werden. Bei deren Wahl (anklicken) öffnet sich das Fenster *Modulerstellung* (**Abbildung 2.1.2**). In diesem müssen Daten für jede Variable eingetragen werden. Dies sind die obere und die untere Grenze jeder Variablen, sprich der Definitionsbereich bzw. der Bereich der zulässigen Werte. Angenommen, der Bereich soll nach oben hin unbegrenzt sein, muss einfach eine utopisch hohe Zahl eingetragen werden. Dazu wird der Startwert der Variable eingegeben und die Startschrittweite. Durch *Apply* wird die Eingabe einer Variablen abgeschlossen, durch *Analyze* die Eingabe der letzten Variablen und gleichzeitig die Modulerstellung.

Im Fenster Optimierung geht die Definition weiter. Dort sind die CAE-Namen einzutragen. Dies sind das CAE-Modell betreffende Namen, um den richtigen job zu starten und um aus der ODB-Datei die richtigen Werte herauslesen zu können. Dann muss der benötigte Ergebniswert definiert werden. Dazu muss das passende Modul ausgewählt werden. Meistens ist die Datei ErgebniswertGewichtMisesSpannung.py,

die das Volumen aller Elemente des Ergebnissets zusammenzählt und als Ergebniswert ausgibt und die größte Spannung aller Elemente als Ergebnis.

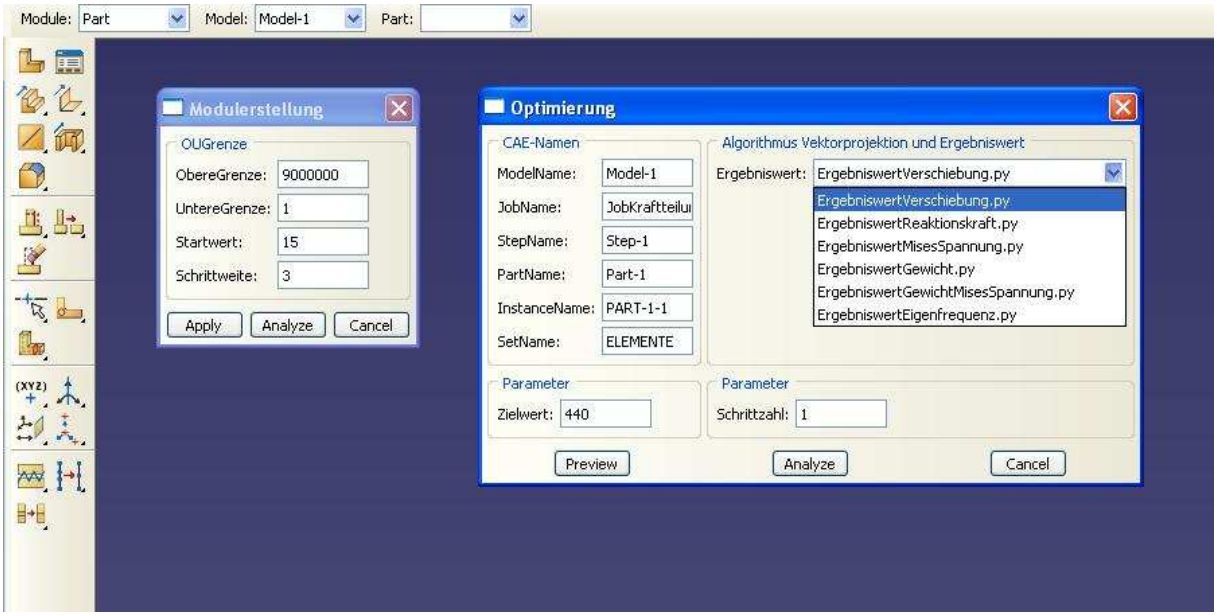


Abb. 2.1.2: Optimierungsmenü im Hauptfenster

Der Parameter Zielwert ist der Wert der Nebenbedingung, z.B. die zulässige Spannung. Die Schrittzahl sagt aus, wie oft bei Nichterfolg die Schrittweite halbiert werden soll, bis das Programm abbricht und damit die Optimierung beendet.

Wenn der Button *Preview* oder *Analyze* geklickt wird, wird die durchzuführende Optimierung als Quellcode geschrieben. Dies Zusammenschreiben ist eine Funktion, die durch Wahl des jeweiligen Algorithmus ausgewählt wurde und nun durch Klicken des Buttons ausgeführt wird. Dieser Quellcode wird in die Datei *Variationsparameter.py* geschrieben und wird durch *Analyze* durch das Kommando *execfile* ausgeführt.

Das Erstellen des Optimierungscodes sieht folgendermaßen aus (**Abbildung 2.1.3**). Zuerst wird ein Deklararationsblock geschrieben, in dem die Variablen des Programms deklariert werden. Dies sind Zählvariablen, Listen und die aus den Eingabemasken übernommenen Variablen und daraus zusammengesetzte Größen. Danach beginnt schon die Optimierungsschleife, die dann durchlaufen wird, bis das Abbruchkriterium erreicht wird. In ihr werden erst den Variablen (die zu variierenden Größen der Optimierung) ihre Startwerte zugeordnet und mit Hilfe von Zählvariablen und Bedingungen die Variablengrößen der Tastschritte aufgeschrieben. Danach folgt der Berechnungs- und Ergebnisblock. Dies ist die Übernahme der vorgefertigten und ausgewählten Module von Lernphase (Variationsmodul.py), Starten der Berechnung (JobStart.py) und Ergebniswert (z.B. Verschiebung.py). Der Rest ist der Auswertungsblock. Dieser ist nicht als Modul vorgefertigt, sondern schreibt aus der

Funktion den Kern des Algorithmus in den Code. Dieser Kern ist der Auswertungsblock, in dem die ermittelten Daten der letzten Schritte erfasst und bewertet werden. Daraus wird im Ablauf der Optimierung entschieden, wohin im Programmcode gesprungen wird und wie die neuen Schritte aussehen.

Deklarationsblock:
Übernahme der Daten aus den Eingabemasken
z.B.: Jobname, Partname, Schrittzahl, Wert der Nebenbedingung,
Zahl der Variablen h, Schrittweiten, Randbedingungen, Startwerte
Optimierungsschleife (while):
Variablen
Berechnungs- und Ergebnisblock:
Aufrufen der im Replay File in Skriptsprache aufgezeichneten Lernphase.
Mit ihr wird das Abaqus/Cae Modell mit den entsprechenden Werten
modifiziert und die Vernetzung durchgeführt.
Starten der Berechnung: mdb.jobs[JobName].submit()
(JobStart.py)
Auslesen des Ergebnisses und des Ergebniswertes (*Ergebniswert*.py)
Übergabe an die Ergebnislisten
Auswertungsblock:
Auswertung der Ergebnisse der Tastschritte
Berechnung des Ausfuhrungsschrittes und gehen zu
mdb.jobs[JobName].submit()
Auswertung der Ergebnisse der Tastschritte
-Wenn die Ergebniswerte nicht verbessert wurden, wird die Schrittweite
halbiert
-Testen der Abbruchbedingung
-Herabsetzen der Zählvariablen x und Index auf Null und dadurch starten
eines neuen Durchlaufes der Optimierungsschleife

Abb. 2.1.3: Aufbau des Moduls Variationsparameter.py

Für die Qualität, die Leistungsfähigkeit und für die Vielseitigkeit des Programms, ist es wichtig, darauf hinzuweisen, dass es variablenunabhängig ist, d.h., es ist unabhängig von der Anzahl an Variablen und nicht auf feste Zahlen festgelegt. So ist es Teil des Programms, in der Deklaration der Optimierung festzustellen, wie groß die Zahl der Variablen ist und davon abhängig den Optimierungscode zu erstellen. Die Zahl der Variablen wird erstellt, indem im Fenster *Modulerstellung* die Variablen einzeln definiert werden. Diese Variablenzahl *h* wird bei der Erstellung des Optimierungscodes abgefragt und umgesetzt. Der Urquellcode der Optimierung, also die Funktion createOptimierung*Algoritmenname* im Modul optiModul.py ist auf diesen

Sachverhalt ausgelegt und ist deshalb mit Schleifen angefüllt, die so aufgebaut sind, dass sie die Bedingung erfüllen: Erfülle die jeweilige Aufgabe, bis x gleich h ist. Dies wird durch einen Auszug aus der Funktion *createOptimierungVektorprojektion* dargestellt.

```
Text22=(' if x==%s or x=='%(h+1)+' %s:' %(h+2))
m=1
x=0
while m<=h:
    wert1=("%s\n")%("        listErgebnisse.append('Vneuneu%s')"%m)
    wert2=("%s\n")%('        listErgebnisse.append(Vneuneu%s)'%m)
    listALALALALEinschubEinschub.append(wert1)
    listALALALALEinschubEinschub.append(wert2)
    m=m+1
```

Abb. 2.1.4: Variablenunabhängigkeit

Bei Text22 wird der Wert von h übernommen und direkt der Code erstellt. Komplizierter ist es in der folgenden Schleife. Dort wird die Schleife solange durchlaufen, bis die Variablenzahl h erreicht ist. Der so entstandene Code ist in **Abbildung 2.1.5** dargestellt.

```
.
        if x==3 or x== 4:
            listErgebnisse.append('Vneuneu1')
            listErgebnisse.append(Vneuneu1)
            listErgebnisse.append('Vneuneu2')
            listErgebnisse.append(Vneuneu2)

.
```

Abb. 2.1.5: Ergebnis von *Variablenunabhängigkeit*

Die im Originalcode von **Abbildung 2.1.4** entstandene Variable Text22 und die Liste listALALALALEinschubEinschub sind Textbausteine, aus denen Variations-parameter.py zusammengesetzt wird. Das funktioniert mit der Liste allerdings nicht. Diese muss erst in einen *string* umgewandelt werden. Damit ist es möglich, das Ausführungsmodul Variationsparameter zu erzeugen. Dazu gibt es den Code am Ende der Funktion C*reateOptimierung* (**Abbildung 2.1.6**). Dort sind die einzelnen Bausteine bzw. Variablen wieder zu erkennen, die bereits erwähnt wurden. Die Variablennamen aus den Fenstern der Optimierung sind zu sehen, der gesamte Komplex Modulerstellung als Variable *ObereUntereGrenze*, die Module des

Berechnungsblocks *Variationsmodul, jobStart* und *Ergebniswert* und die variabel erstellbaren Teile des Auswertungsblocks *Text1, Text2,* usw.. Diese werden also aneinandergereit und mit Datei2.write in das Modul *Variationsparameter.py* geschrieben.

```
....

werte=("""%s%s\n%s%s\n%s'%s'\n%s'%s'\n%s'%s'\n%s'%s'\n%s'%s'
        %s'%s'\n%s'%s'\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s
        %s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s
        %s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s""")%(
    'Zielwert = ',Zielwert,'Schrittzahl = ',Schrittzahl,\
    'ModelName =',ModelName,'JobName =',JobName,'StepName =',StepName,
    'PartName=',PartName,'InstanceName =',InstanceName,'SetName =',\
    SetName,  'ErgebniswertName =',ErgebniswertName,JobNa,BilderOrdner,\
    NeuerOdbName,'list=[]',ObereUntereGrenze,AL,ALA,text0,Bedingung1,\
    Bedingung11,ALEinschub,ALAL,Bedingung2,ALALALA,ALEinschub2,Text4,\
    ALALALAL,Text5,ALALALAEinschub,Text7,ALALALALEinschub2,ALALALALEinsch
    Text9,Variablenausgabe,VariationsModul,jobStart,Ergebniswert,\
    TextLenListErgebnisse,Text1,ALALA,Text2,ALALALALALAEinschub,Text22,\
    ALALALALEinschubEinschub,ALALAL,Text3,Text10)

Datei2.write(werte)
Datei2.close()
```

Abb. 2.1.6: Organisation der Modulerstellung

3. Algorithmen

Der Begriff des Algorithmus wird dem Gebiet der praktischen Mathematik zugeordnet. Darunter wird ein Satz von Regeln verstanden, um ein Problem schrittweise zu lösen [Bai94]. Beispiele für einen Algorithmus sind das allgemeine Verfahren der Divisionsrechnung und der Gauss-Algorithmus, bei dem ein inhomogenes lineares Gleichungssystem mit n Variablen und n unabhängigen Gleichungen durch schrittweise Elimination der Variablen in ein gestaffeltes (Dreiecks-) System überführt wird und somit gelöst werden kann. Ein iterativer Algorithmus ist ein Prozess, bei dem ein systematisches Verfahren wieder und wieder durchgeführt wird, bis eine Lösung erreicht ist. Den gesamten Durchlauf dieses Verfahrens nennt man Iteration [Hil88]. Die Algorithmen dieser Arbeit sind iterative Algorithmen.

Das Prinzip des Optimierungsalgorithmus wurde schon in Kapitel 1 behandelt und in der dort geschilderten Weise auch in dieser Arbeit umgesetzt. Dies bedeutet, das man von einem oder mehreren Startwerten ausgehend nach bestimmten Regeln schrittweise eine Verbesserung zu erreichen versucht, bis man ein bestimmtes vorher eingegebenes Abbruchkriterium erfüllt. In diesem Kapitel werden die entwickelten Algorithmen dieser Arbeit vorgestellt und ihre theoretischen Prinzipien und Arbeitsweisen erläutert.

3.1. Der Algorithmus „Halbe Schrittweite“

Die prinzipielle Arbeitsweise bzw. Programmierung des Algorithmus *Halbe Schrittweite* wird in **Abbildung 3.1.1** dargestellt. Im Deklarationsblock werden Typen wie Zahlen, Strings oder Listen initiiert. Dazu gehören die Ergebnislisten, in denen relevante Daten der Optimierung, wie Variablenwerte oder Ergebniswerte der Zielfunktion dokumentiert werden oder es werden Daten temporär gespeichert, um diese zu weiterführenden Berechnungen heranzuziehen. Weiterhin gehören zu den Typen des Deklarationsblocks Zählvariablen, Variationsparameter, Randbedingungen oder Variablen wie *Jobname* oder *Partname*. Die beiden Letztgenannten zum Beispiel werden benötigt, um direkt auf CAE zuzugreifen und dort die richtige Berechnung (job) zu starten, oder aus der Ausgabedatei (odb File) die richtigen Daten auszulesen. Nach der Deklaration oder der Initiierung werden den Variablen die Werte oder die entsprechenden Namen zugeordnet. Diese werden aus

den Eingabemasken übernommen. Die Optimierungsschleife ist eine Endlosschleife. und wird solange durchlaufen, bis die Abbruchbedingung erreicht ist und durch eine break-Anweisung die Schleife verlassen wird. Die Variablen V_{oben1} und V_{unten1} sind die obere und untere Grenze des Variationsparameters $Variable_1$ oder Var_1. An ihnen wird immer die Zulässigkeit des geplanten Schrittes überprüft und die Tastschritte durchgeführt. Dies geschieht natürlich nacheinander. Das heißt, dass zuerst die Variablen für den Tastschritt 1 festgelegt werden. Diese Werte werden dann dem Berechnungs- und Auswertungsblock übergeben und mit ihnen wird das Cae-Modell modifiziert und die Berechnung wird gestartet.

Danach werden die Ergebniswerte ausgelesen und aus diesen Erkenntnissen die neuen Variablenwerte festgelegt. Unter der Annahme, dass der erste Tastschritt das beste Ergebnis gebracht hat, also der Index = 1 ist, wird die Testannahme für Var_1 verifiziert, die Zählvariablen werden auf Null gesetzt und mit dem neuen Wert für den $Variationsparameter_1$ wird die Optimierungsschleife neu durchlaufen.

In **Abbildung 3.1.2** ist der Algorithmus skizziert. Dies ist die Darstellung des Bewegungsprofils eines zweidimensionalen Optimierungsproblems. Aus der Eingabemaske werden also für jede Variable die obere und untere Grenze übernommen ($V_{oben,i}$, $V_{unten,i}$). Diese Werte bilden die Randbedingung. Von diesem Punkt aus ergeben Rechenschritte, die jeder Variablen die jeweils eingegebene Schrittweite abzieht beziehungsweise dazurechnet, Ergebnisse, aus denen der beste Wert den neuen Ausgangspunkt darstellt.

Würde eine Variable die Randbedingung übertreten, wird diese Addition nicht ausgeführt. Der Variablenwert bleibt unverändert. Übertritt der Ergebniswert einer Rechnung die Nebenbedingung, wird dieses Ergebnis nicht zur Bewertung der Einzelergebnisse zugelassen. Ergeben die Einzelergebnisse während eines Testzyklus keine Verbesserung, wird der Testzyklus mit halbierten Schrittweiten wiederholt. Die Zahl der Schrittweitenhalbierungen ist als Parameter einzugeben. Der Name „Halbe Schrittweite“ des Algorithmus bedeutet, dass bei Erreichen eines bestimmten Kriteriums die Schrittweite des Algorithmus halbiert wird und so die weitere Annäherung an das Optimum ermöglicht wird.

Deklarationsblock:
Übernahme der Daten aus den Eingabemasken
z.B.: Jobname, Partname, Schrittzahl, Wert der Nebenbedingung,
Zahl der Variablen h, Schrittweiten, Randbedingungen, Startwerte

Optimierungsschleife:
while x<h+1:
    Tastschritte:
    if x=1 and Var_1+$Schrittweite_1$<=V_{oben1}:
        $Variable_1$ ⟵ Var_1+$Schrittweite_1$
        $Variable_2$ ⟵ Var_2
    if x=2…….

Berechnungs- und Ergebnisblock:
Aufrufen der im Replay File in Skriptsprache aufgezeichneten Lernphase.
Mit ihr wird das Abaqus/Cae Modell mit den entsprechenden Werten
modifiziert und die Vernetzung durchgeführt.
Starten der Berechnung: mdb.jobs[JobName].submit()
Auslesen des Ergebnisses und des Ergebniswertes
    if Ergebniswert<ErgebniswertAlt and Ergebnis<=Zielwert:
        ErgebniswertAlt=Ergebniswert
        Index=x
        Gesamtindex=r
Übergabe an die Ergebnislisten

Auswertungsblock:
Auswertung der Ergebnisse der Tastschritte
    if x=4:
        if Index=1 and Var_1+$Schrittweite_1$<=V_{oben1}:
            Var_1 ⟵ Var_1+$Schrittweite_1$
        if Index=2 and Var_1-$Schrittweite_1$>=V_{unten1}:……..
        (Der Index zeigt an, welcher Tastschritt das beste Resultat
        erbracht hat)
-Wenn die Ergebniswerte nicht verbessert wurden, wird die Schrittweite
 halbiert (nach der Schrittzahl der Eingabemaske)
-Testen der Abbruchbedingung
-Herabsetzen der Zählvariablen x und Index auf Null und dadurch starten
 eines neuen Durchlaufes der Optimierungsschleife

Abb. 3.1.1: Prinzipieller Aufbau des Algorithmus Halbe Schrittweite

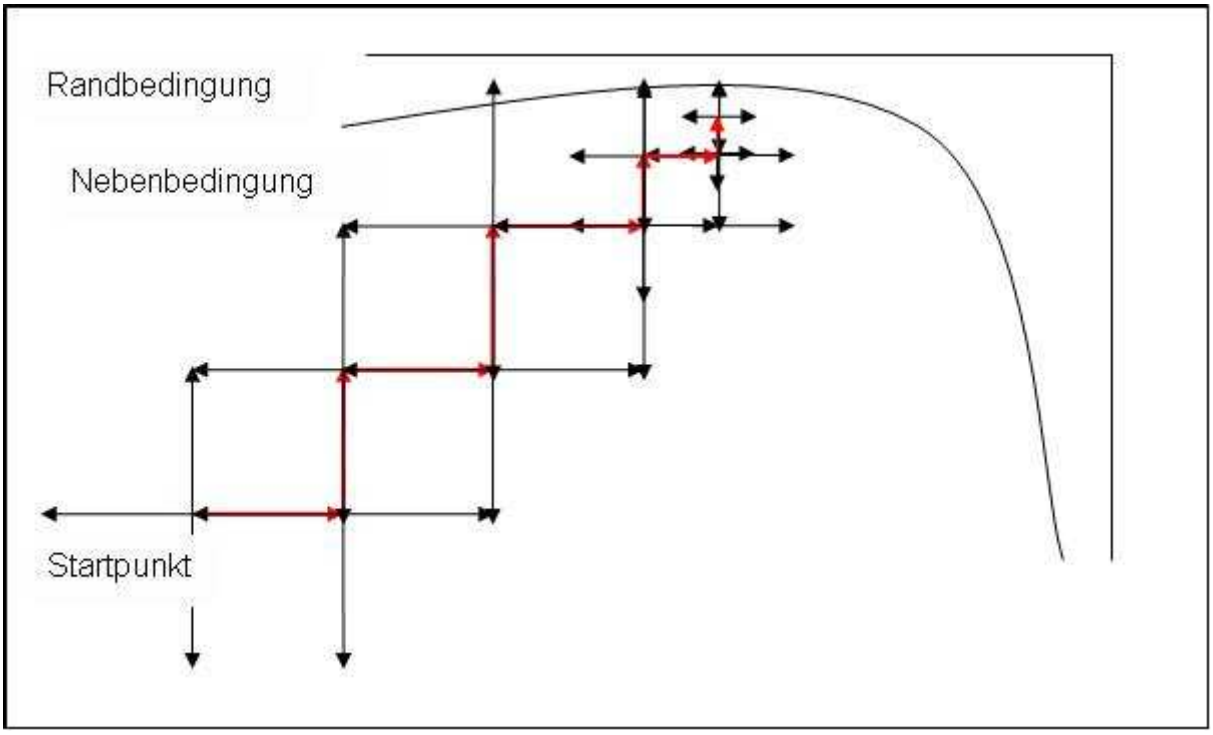


Abb. 3.1.2: Skizze des Algorithmus Halbe Schrittweite

3.2. Der Algorithmus „Gradient“

Auch der Algorithmus Gradient (**Abbildung 3.2.1)** beginnt wie die Halbe Schrittweite nach der Definition des Startwertes mit Tastschritten in beide Richtungen sämtlicher Variablen. Nur ist nun nicht der beste Einzelwert das Ergebnis des Testzyklus, sondern ein aus den Werten ermittelter Gradient. Der Begriff Gradient ist in diesem Zusammenhang nicht ganz korrekt, da nicht das totale Differential an dieser Stelle ermittelt wird. Vielmehr werden die Schrittweiten entsprechend der prozentualen Verbesserung der Zielfunktion, die sie erbringen, gewichtet. So bekommt der beste Wert die Gewichtung = 1, d.h. die zugehörige Variable wird mit der Schrittweite des Tastschrittes verändert, während auf die anderen Schrittweiten die prozentuale Verbesserung angerechnet wird (Werte von null bis eins). Auf diese Weise entsteht ein Vektor, der so genannte Gradient.

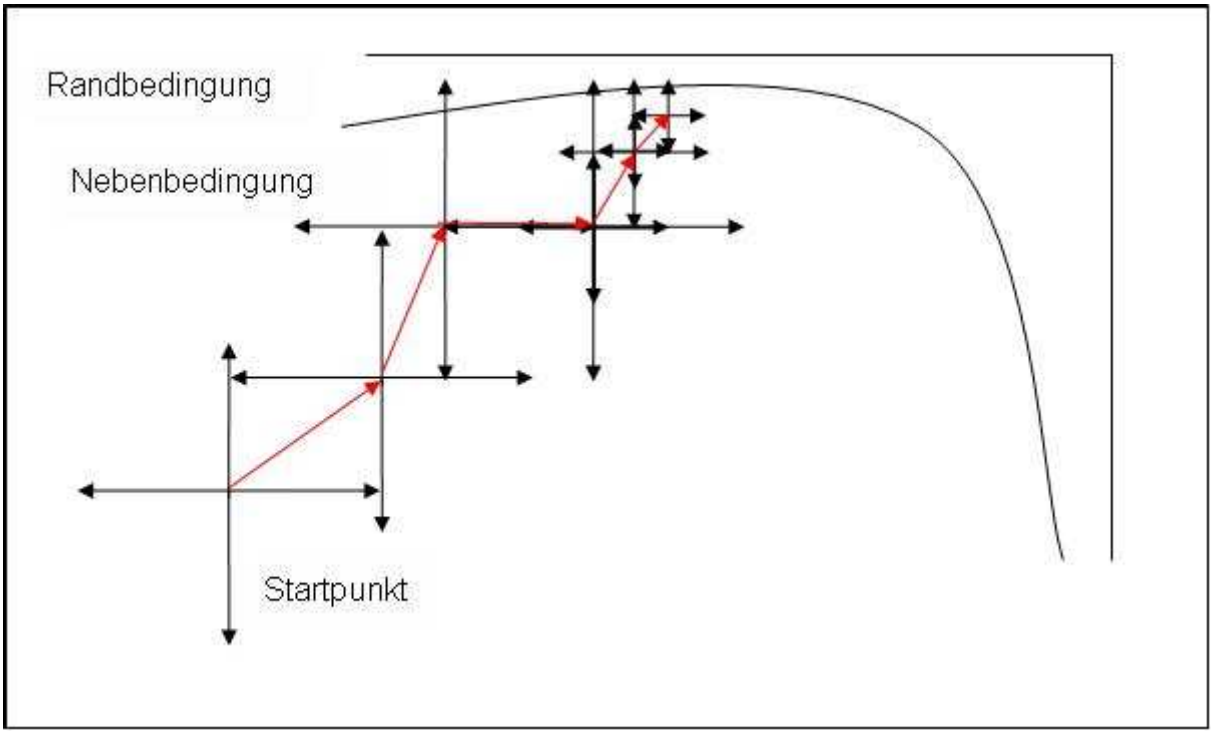


Abb. 3.2.1: Skizze des Algorithmus Gradient

Erzielt eine Variable in beide Richtungen keine Verbesserung, wird sie nicht mit einbezogen. Würde ein Testschritt die Randbedingung übertreten, würde mit dem Ausgangswert dieser Variable gerechnet. Übertritt ein Tastschritt die Nebenbedingung, so wird diese Variable aus der Gradientenermitlung herausgenommen. Wie aus der Skizze des Algorithmus Gradient zu ersehen ist, erfolgt bei Fehlen des Erfolges bei einem Testzyklus die Wiederholung auch hier mit halber Schrittweite.

Der Algorithmus in dieser Form hat allerdings den Nachteil, dass in seltenen Fällen der Ausführungsvektor die Nebenbedingung verletzt. Dann besteht die Gefahr, dass der Algorithmus dort „hängen bleibt“. Für diesen Fall gibt es die modifizierte Version *Gradient-Zurück.* Dieser hat die Möglichkeit festzustellen, dass die Nebenbedingung verletzt wurde und wird dann die Länge des Ausführungsvektors solange halbieren, bis er sich wieder im zulässigen Raum befindet. Dann wird mit halbierter Schrittweite der Tastzyklus wieder aufgenommen (**Abbildung 3.2.2)**.

Deklarationsblock:
Übernahme der Daten aus den Eingabemasken

Optimierungsschleife:
while x<h+1:
    if x=1 and Var_1+$Schrittweite_1$<=V_{oben1}:
        $Variable_1$ ⟵ Var_1+$Schrittweite_1$
        $Variable_2$ ⟵ Var_2
    if x=1 and Var_1+$Schrittweite_1$>V_{oben1}:
        $Variable_1$ ⟵ Vb1
        $Variable_2$ ⟵ Var_2
Berechnungs- und Ergebnisblock:
Starten der Berechnung: mdb.jobs[JobName].submit()
Auslesen des Ergebnisses und des Ergebniswertes
Übergabe an die Ergebnislisten

Auswertungsblock:
Auswertung der Ergebnisse der Tastschritte
    if x == Zahl der Variablen +1:
        Ermittlung des jeweils besseren Ergebnisses E_i der Variablen V_i
        Daraus wird der Richtungssinn jeder V_i bestimmt und die
        Gewichtung jeder V_i berechnet:
        Gewichtung (Δ Emax) ⟵ 1
        Gewichtung (Δ E_i) ⟵ Δ E_i / Δ Emax
        Neuer Vektor (Ausführungsvektor):
        F ⟵ $Richtungssinn_i$ *$Gewichtung_i$ *$Schrittweite_i$
-Wenn die Ergebniswerte durch den Tastschrittzyklus nicht verbessert wurden, wird die Schrittweite halbiert (nach der Schrittzahl der Eingabemaske)
-Testen der Abbruchbedingung
-Herabsetzen der Zählvariablen x und Index auf Null und dadurch starten eines neuen Durchlaufes der Optimierungsschleife

Abb. 3.2.2: Prinzipieller Aufbau des Algorithmus Gradient

3.3. Der Algorithmus „Gradient-Nebenbedingung“

Aus dem Algorithmus Gradient sei eine Variation, der Algorithmus Gradient-Nebenbedingung, generiert. Dabei bedeutet der Name nicht, dass eine Nebenbedingung behandelt werden kann, sondern der Algorithmus ist darauf

ausgelegt, schnell den Rand des Raumes der Nebenbedingung zu erreichen, da auf ihm das Optimum zu erwarten ist.

Die Funktionsweise ist prinzipiell zu der des Algorithmus Gradient identisch (**Abbildung 3.3.1**). Dies betrifft die Gradientenermittlung über einen Tastzyklus und das Verhalten an Rand- und Nebenbedingungen. Der Unterschied zeigt sich nach dem ersten Testzyklus. Aus diesem Zyklus wird ein Vektor ermittelt und ein Schritt in dessen Richtung ausgeführt. Darauf folgt allerdings kein weiterer Testzyklus, sondern der ermittelte Vektor wird nochmals mit gleicher Schrittweite von der Spitze des ersten Vektors als Schritt ausgeführt.

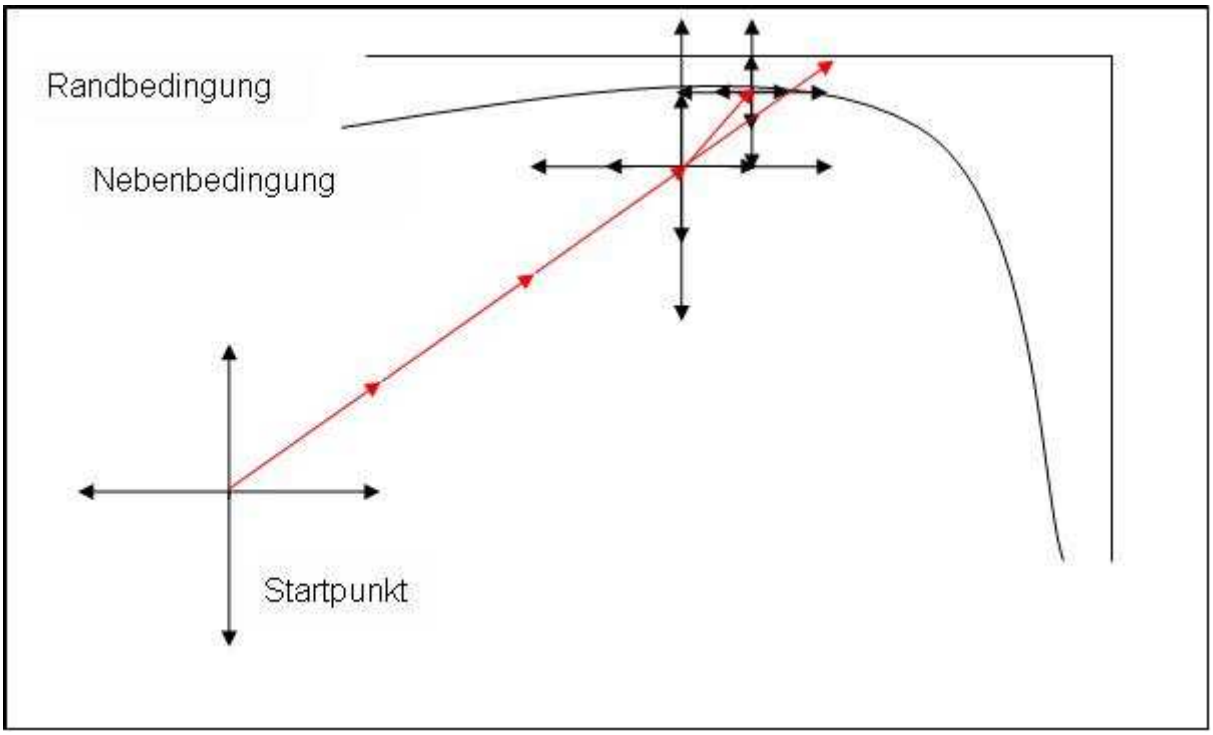


Abb. 3.3.1: Skizze des Algorithmus Gradient-Nebenbedingung

Diese Schrittwiederholung wird unterbrochen, wenn der Ergebniswert keine Verbesserung aufweist. Dann wird ein neuer Gradient ermittelt. Wenn in einem Ausführungsschritt die Randbedingung verletzt wird, geht der Algorithmus einen Schritt zurück. Von dem letzten zulässigen Punkt aus wird ein neuer Vektor ermittelt, gegebenenfalls mit halber Schrittweite.

Wird in einem Tastschritt die Randbedingung verletzt, wird nicht mit diesem Wert der Variablen gerechnet werden, sondern die Variable wird verkürzt und es wird der Wert der Randbedingung als Variablenwert eingesetzt und an dieser Stelle der Ergebniswert der Zielfunktion ermittelt. In einem Ausführungsschritt ist diese Möglichkeit nicht gegeben. Dies ist eine Verletzung der Ausführungsbedingung, was zu Folge hat, dass ein Schritt zurückgegangen wird und ein neuer Vektor ermittelt wird.

An dieser Stelle sollen noch die Algorithmen *Gradient-Lang* und *Gradient-Nebenbedingung-Lang* erwähnt werden. Diese sind mit einer Ausnahme identisch zu *Gradient* bzw. *Gradient-Nebenbedingung*. Ihre Gewichtungen werden auf beide Δ E der jeweiligen Variablen normiert und nicht nur auf eine Änderungslänge.

3.4. Der Algorithmus „Goldener Schnitt“

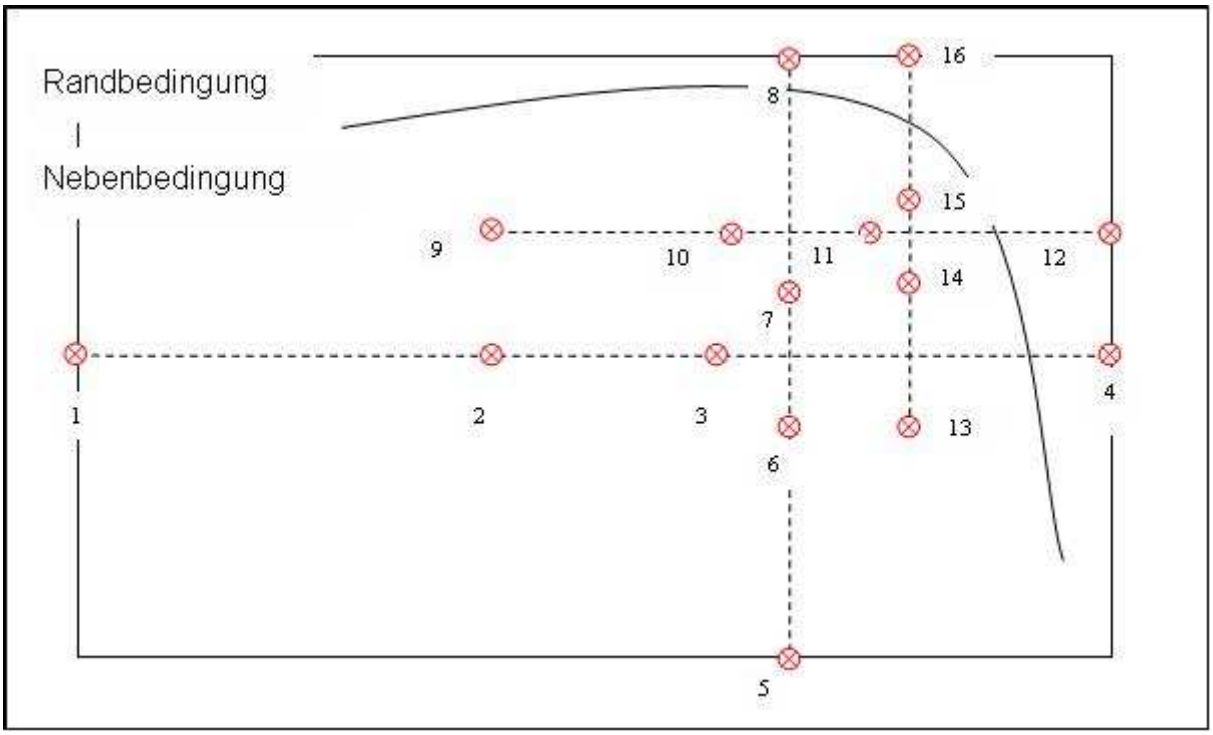


Abb. 3.4.1: Skizze des Algorithmus Goldener Schnitt

Der Goldene Schnitt unterscheidet sich im Aufbau von den anderen Algorithmen (**Abbildung 3.4.1**). Es werden zwar auch die obere und untere Grenze für jede Variable eingegeben, aber ohne eine Schrittweite angeben zu müssen, wird der gesamte Definitionsbereich unterteilt. Aus der Differenz von oberer und unterer Grenze wird für jede Variable die Mitte zwischen den Grenzen bestimmt. Während diese Werte für die Variablen konstant gehalten werden, wird eine der Variablen variiert. Die Werte für die erste und die vierte Rechnung sind die der Randbedingungen. Die zweite und dritte Rechnung sind jeweils im Abstand von 0.618 von den Rändern. Nach den ersten vier Rechnungen findet eine Bewertung statt. Angenommen, der Ergebniswert von Rechnung 3 ist niedriger als der von 2, verschiebt sich die linke Randbedingung auf den Variablenwert von Rechnung 2. Der neue konstante Wert dieser Variable ist die Mitte zwischen Wert 2 und Wert 4. Bei konstanter Variable 1 werden die Rechnungen 5 – 8 durchgeführt.

Sollte der bessere innere Wert außerhalb der Randbedingung liegen, werden aus diesem Rechenlauf die Grenzen nicht verschoben. Der Algorithmus arbeitet aber die eingegeben Schrittzahl bis zum Ende ab.

In **Abbildung 3.4.2** ist der prinzipielle Aufbau des Algorithmus *Goldener Schnitt* dargestellt. Dieser unterscheidet sich von den anderen Algorithmen, dass er zwei innere Schleifen nutzt, in denen jeweils der Berechnungs- und Auswertungsblock mit der gesamten Lernphase aufgerufen wird. Dies verlängert den Quellcode und macht ihn unübersichtlicher, hat aber den Vorteil, dass er in der Möglichkeit, Entscheidungsverzweigungen abzuarbeiten, vergrößert wird. Dies ist für weitere Arbeiten wichtig, bei denen zum Beispiel die einzelnen Algorithmen zu einer selbst ablaufenden Strategie zusammengefasst würden. Die hier weiter vorgestellten Algorithmen benötigen dies jedoch noch nicht, und sind trotz ihrer steigenden Komplexität, vom Prinzip her der bisher beschriebenen Algorithmengruppe *Gradient* ähnlich.

Die bisherigen Algorithmen wurden im Zusammenhang mit einer Nebenbedingung, dem Erreichen der zulässigen Spannung, dargestellt. Dies ist die Nebenbedingung, die in dieser Phase dem Quellcode vorliegt. Prinzipiell gelten die programmierten Voraussetzungen auch für andere Randbedingungen, die ohne sehr großen Aufwand in den Code eingepflegt werden könnten. Diese Algorithmen sind aber auch für unrestringierte Aufgaben einsetzbar. Sollte in der Programmeingabemaske nicht das Ergebnismodul *Ergebniswert-Gewicht-Misesspannung* gewählt werden, sondern *Verschiebung* oder *Kontaktkraf*t, würde das Programm nur auf eine der beiden letztgenannten Größen hin optimieren, ohne eine Randbedingung mit einzubeziehen. Die in den Bildern und Text beschriebenen Algorithmen behalten also ihre prinzipielle Funktionsweise. Nur wird nicht mehr das Volumen bzw. das Gewicht minimiert, sondern die anderweitig gewählte Ergebniswertgröße, und die Überprüfung, ob die Randbedingung eingehalten wurde findet nicht mehr statt. Dies gilt für alle bisher vorgestellten Algorithmen

Weiterhin ist zu den bisherigen Algorithmen zu sagen, dass ihre Sensitivitätsanalyse und deren Auswertung bzw. die Bestimmung des Ausführungsvektors in einer bestimmten Art und Weise vonstatten geht. So werden die einzelnen Variablen nacheinander abgearbeitet und jede Variablenänderung für sich eine Verbesserung bringen muss, um umgesetzt zu werden. Dies hat zur Folge, dass die Richtungen der Algorithmen beschränkt werden, was im folgenden Kapitel an 2-dimensionalen Beispielen anschaulich gemacht werden wird. Die im Folgenden dargestellten Algorithmen sind so konzipiert, das sie dieser Beschränkung nicht mehr unterliegen.

Deklarationsblock:
Übernahme der Daten aus den Eingabemasken
Äußere Optimierungsschleife:
while u<Schrittzahl:
    Definition der Rechnungswerte
    $s \leftarrow 0.618*(V_{b1}-V_{a1})$
    $V_{c1} \leftarrow V_{a1}+s$
    $V_{d1} \leftarrow V_{b1}-s$
    $list_1 \leftarrow [V_{a1},V_{d1},V_{c1},V_{b1}]$
    $s \leftarrow 0.618*(V_{b2}-V_{a2})$
    $V_{c2} \leftarrow V_{a2}+s$
    $V_{d2} \leftarrow V_{b2}-s$
    $list_2 \leftarrow [V_{a2},V_{d2},V_{c2},V_{b2}]$
    Innere Schleifen
     while i<4:
        Ausführen der Werte
        $Variable_1 \leftarrow list_1[i]$
        $Variable_2 \leftarrow (V_{a2}+(V_{b2}-V_{a2})/2)$
        Berechnungs- und Ergebnisblock:
        Starten der Berechnung: mdb.jobs[JobName].submit()
        Auslesen des Ergebnisses und des Ergebniswertes
        Übergabe an die Ergebnislisten
        Auswertungsblock:
        Verschieben der Randwerte und ermitteln des neuen Mittelwertes
     while i<4:
        Ausführen der Werte
        $Variable_2 \leftarrow list_2[i]$
        $Variable_1 \leftarrow (V_{a1}+(V_{b1}-V_{a1})/2)$
        Berechnungs- und Ergebnisblock:
        Starten der Berechnung: mdb.jobs[JobName].submit()
        Auslesen des Ergebnisses und des Ergebniswertes
        Übergabe an die Ergebnislisten
        Auswertungsblock:
        Verschieben der Randwerte und ermitteln des neuen Mittelwertes

Abb. 3.4.2: Prinzipieller Aufbau des Algorithmus *Goldener Schnitt*

3.5. Der Algorithmus „Vektorprojektion“

Ein Qualitätsmerkmal der folgenden Algorithmen ist die Fähigkeit, eine Variable zu „verschlechtern“, wenn in Verbindung mit der Änderung einer oder mehrerer Variablen das Gesamtergebnis verbessert wird. Dies ermöglicht theoretisch das Erreichen des Optimums von jedem Startpunkt aus, vorausgesetzt, er befindet sich im zulässigen Raum.

Der Algorithmus *Vektorprojektion* hat nun eine ganz bestimmte Spezifikation. Er ist darauf spezialisiert, an der Hyperraumebene der Restriktion „entlangzugehen“, an der das Optimum zu erwarten ist. Für den 2-dimensionalen Fall ist dies eine Linie. Da die hier behandelte Restriktion die zulässige Spannung ist, wird im Folgenden von der Spannungslinie oder der Spannungsfläche gesprochen. Dabei halten sich die Algorithmen streng an die mathematische Definition des Gradienten.

Der Gradient eines Skalarfeldes $\Phi = \Phi(x,y,z)$ ist das Produkt aus dem Nabla-Operator ∇ und einem Skalar Φ.

$$grad\,\Phi = \nabla\Phi = \begin{pmatrix} \frac{\partial\Phi}{\partial x} \\ \frac{\partial\Phi}{\partial y} \\ \frac{\partial\Phi}{\partial z} \end{pmatrix} \qquad \text{Gl. 3.1}$$

Die Eigenschaften des Gradienten eines Skalarfeldes sind, dass der Gradient senkrecht auf den Niveaulinien von Φ steht und in Richtung des größten Zuwachses von Φ zeigt. Somit zeigt der negative Volumengradient eines Feldes in Richtung der größten Verkleinerung des Wertes des Feldes.

Da aber das Skalarfeld als Funktion nicht bekannt ist, kann deshalb auch nicht die Ableitung gebildet werden. So müssen die jeweiligen Steigungen des Feldes in den relevanten Punkten jeweils einzeln ermittelt werden. Dazu nutzt man die Definition der Ableitung, bei der die Grenzwerte an einer Stelle gebildet werden. Aus praktischen Gründen und da bei einer reellen Messung keine infinitesimalen Schrittweiten, sondern endliche Schrittweiten genutzt werden müssen, wird in diesem Fall nicht der Grenzwert und damit die Tangente in einem Punkt gebildet, sondern die Sekante. Die Ableitungen werden über das Differenzenverfahren angenährt (Gl. 3.2).

$$\frac{\partial \Phi}{\partial x} = \frac{\Phi(x+\Delta x; y) - \Phi(x; y)}{\Delta x}$$

$$\frac{\partial \Phi}{\partial y} = \frac{\Phi(x; y+\Delta y) - \Phi(x; y)}{\Delta y}$$

Gl. 3.2

Weiter unten ist die Entstehung des Volumengradienten im skalaren Volumenfeld durch das Differenzenverfahren skizziert. Die Länge der Tastschritte sind Δx und Δy. An diesen Punkten erhält man die Ergebniswerte V_1 und V_2. Zieht man von diesen den Wert V ab und teilt diese Werte durch die Schrittlängen, erhält man den Gradienten $\vec{v}$.

$$\Delta V_1 = \frac{\Delta V_1'}{\Delta x_1}$$

$$\Delta V_2 = \frac{\Delta V_2'}{\Delta y_2}$$

$$\vec{v} = \begin{pmatrix} \Delta V_1 \\ \Delta V_2 \end{pmatrix}$$

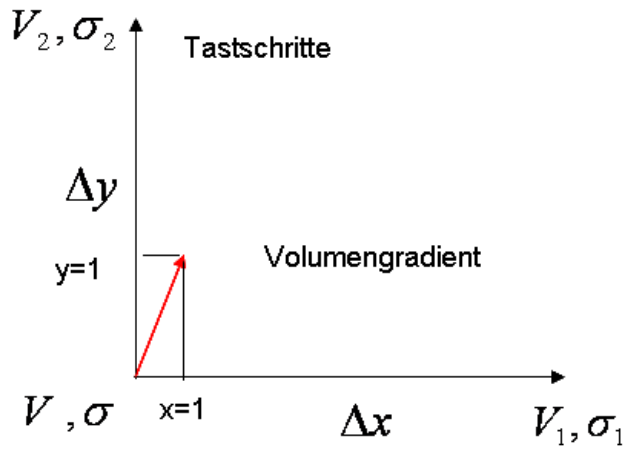


Abb. 3.5.1: Gradient

Die Bestimmung der Länge des Gradientenvektors wird in **Abbildung 3.5.2** dargestellt und beginnt damit, dass überprüft wird, welche Variable welchen Einfluss auf das Ergebnis hat. Zum Beispiel, wenn nach dem größten Einfluss gefragt ist, findet folgende Überprüfung statt:

$$Wenn\, \Delta V_2 > \Delta V_1\, dann\, \Delta V = \Delta V_2\, und\, h = \Delta y$$

Mit dieser Information wird der Gradient auf ΔV normiert und mit der Schrittweite wird die Länge des Gradientenvektors gesteuert. Dazu wird er um 180° gedreht, um nicht in Richtung des größten Zuwachses, sondern in Richtung Minimierung zu zeigen.

$$\vec{v} = \begin{pmatrix} -\Delta V_1 / \Delta V \\ -\Delta V_2 / \Delta V \end{pmatrix} * h = \begin{pmatrix} -a \\ -1 \end{pmatrix} * h$$

Gl. 3.3

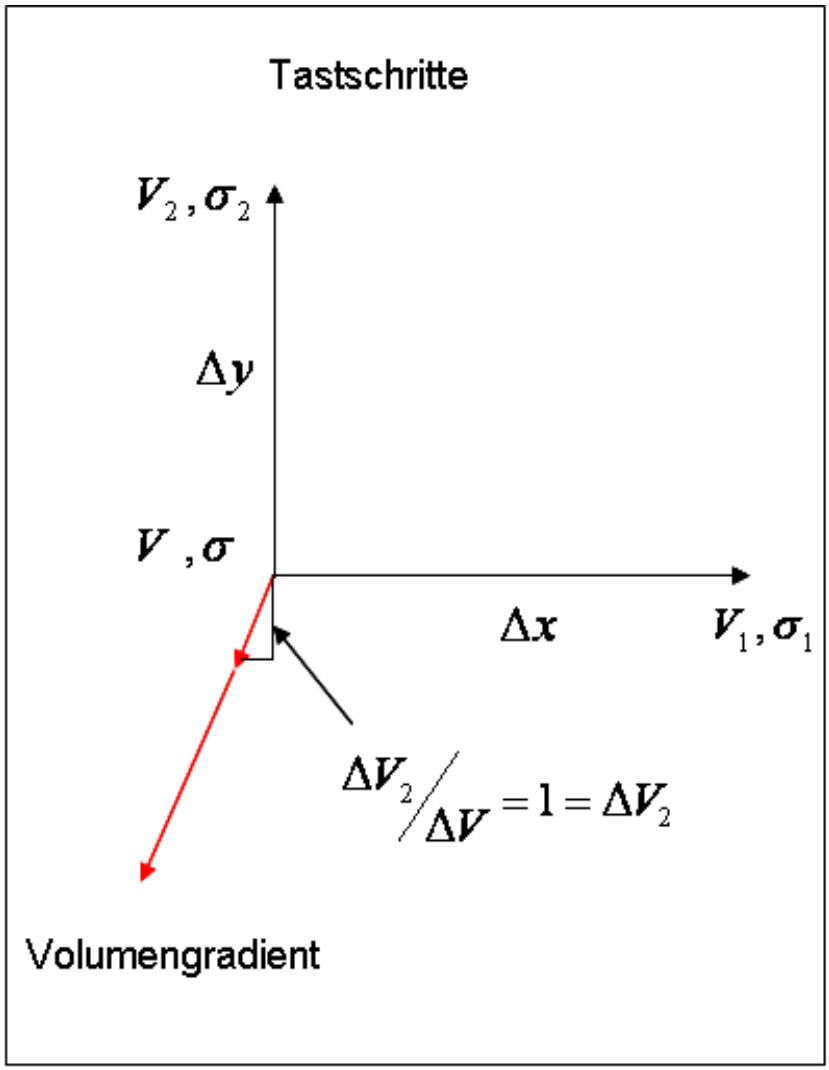


Abb. 3.5.2: Schrittlänge des Gradienten

Der Algorithmus Vektorprojektion funktioniert in der Weise, dass er das Optimum direkt an der Sigmalinie sucht. Dazu ermittelt er neben dem Volumengradienten des Volumenfeldes, den Spannungsgradienten des Spannungs- oder des Sigmafeldes. Der nächste Schritt ist der Namensgeber des Algorithmus. Nun wird der Volumengradient auf den Spannungsgradienten projiziert (Gl. 3.4). Dieser projizierte Vektor kann als Richtungsvektor verschoben werden und muss dann nur noch von dem Volumengradienten abgezogen werden. So erhält man als Ausführungsschritt einen Vektor, der tangential zur Sigmalinie in Richtung einer Verbesserung zeigt (**Abbildung 3.5.3**).

Projektion eines Vektors $\vec{b}$ auf einen Vektor $\vec{a}$.

$$\vec{b}_a = \left(\frac{\vec{a} * \vec{b}}{|\vec{a}|^2} \right) \vec{a} \qquad \text{Gl. 3.4}$$

$\vec{b}_a$ ist die Komponente des Vektors $\vec{b}$ in Richtung des Vektors $\vec{a}$

Der Ausführungsschritt wird folgendermaßen errechnet:

Ausführungsschritt: $\vec{S}$

Volumengradient: $\vec{v}$

Projektion: $\vec{b}_a$

$$\vec{S} = \vec{v} - \vec{b}_a \qquad \text{Gl. 3.5}$$

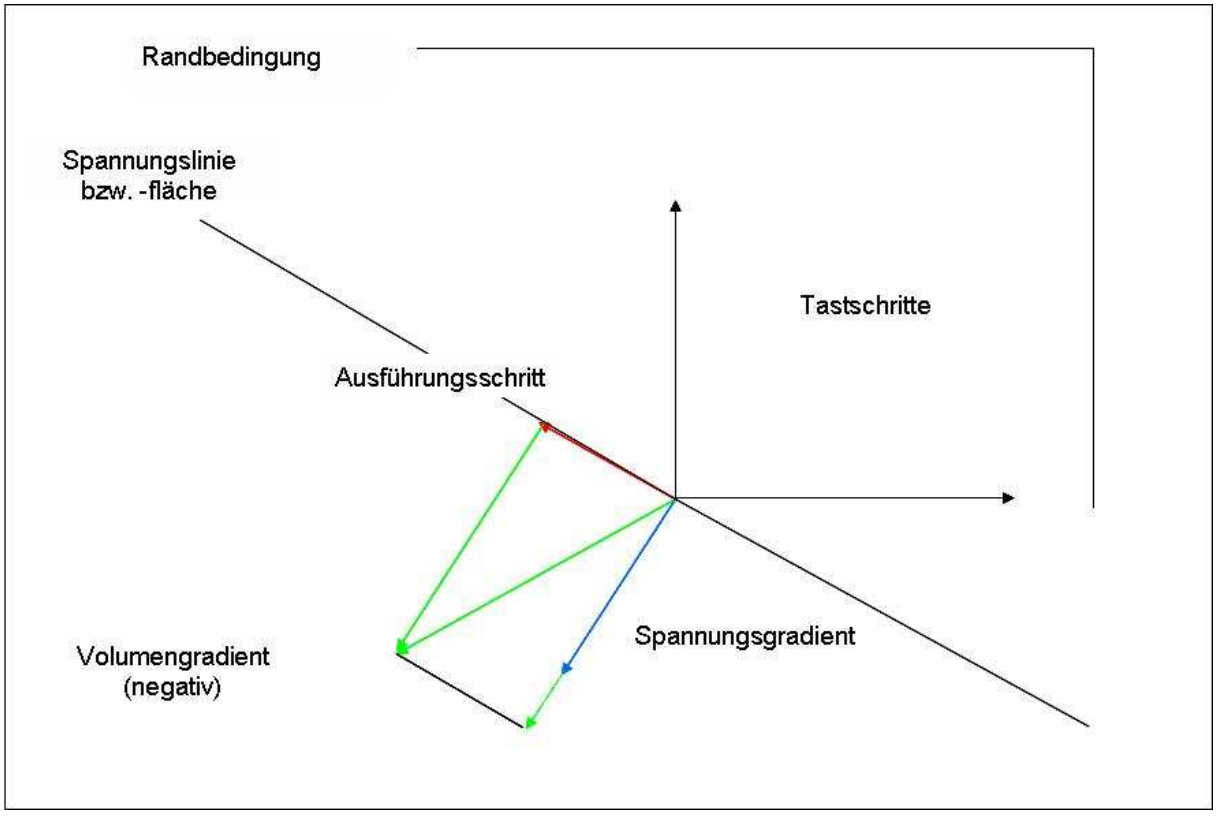


Abb. 3.5.3: Skizze des Algorithmus Vektorprojektion

So wie die Situation in **Abbildung 3.5.3** aufgezeigt wird, ist die Spannungslinie als Äquipotentiallinie zu betrachten. Dass der Startwert genau auf der Sigmagrenzlinie liegt, ist eher unwahrscheinlich. Weiter ist zu dieser Ausbildung des Algorithmus zu sagen, dass er enge Grenzen für seine Einsatzmöglichkeit aufweist. Angenommen die Sigmalinie verläuft konvex, was häufig der Fall ist, wird der Algorithmus

wahrscheinlich irgendwann in den nichtzulässigen Bereich „laufen“ und ihn nicht mehr verlassen können. Aus diesem Grund musste der Algorithmus erweitert werden. Diese Erweiterung nennt sich Sigmakorrektur und ist in **Abbildung 3.5.4** dargestellt.

Der Algorithmus *Vektorprojektion-Sigmakorrektur* arbeitet, wie der Algorithmus Vektorprojektion nur zufälligerweise auf der Sigmalinie. Viel wahrscheinlicher ist, dass er mit einer Äquipotentiallinie in unmittelbarer Nähe zur Sigmalinie arbeitet. Neu ist seine Fähigkeit auf den nichtzulässigen Bereich mit der Sigmakorrektur zu reagieren. In dem Fall wird der schon ermittelte Spannungsgradient genutzt, der senkrecht auf der Spannungslinie steht. Dieser wird um 180° gedreht und an die Spitze des Ausführungsschrittes gesetzt. Dieser Schritt wird solange wiederholt, bis der zulässige Bereich erreicht ist. Wenn dann noch kein verbesserter Volumenergebniswert erreicht ist, wird der Sigmagradient solange wiederum gedreht und dabei verkürzt, bis ein besserer Wert erreicht wurde. Tritt auf diesem Wege nach einer vorher eingegebenen Zahl von Versuchen keine Verbesserung ein, geht der Algorithmus zurück und wiederholt die Gradientenermitlung und Vektorprojektion mit verkürzter Schrittweite und versucht auf diese Weise, sich dem Optimum weiter zu nähern. Die Vermutung liegt dann nahe, dass man mit der alten Schrittweite schon an dem Optimum vorbeigezogen ist.

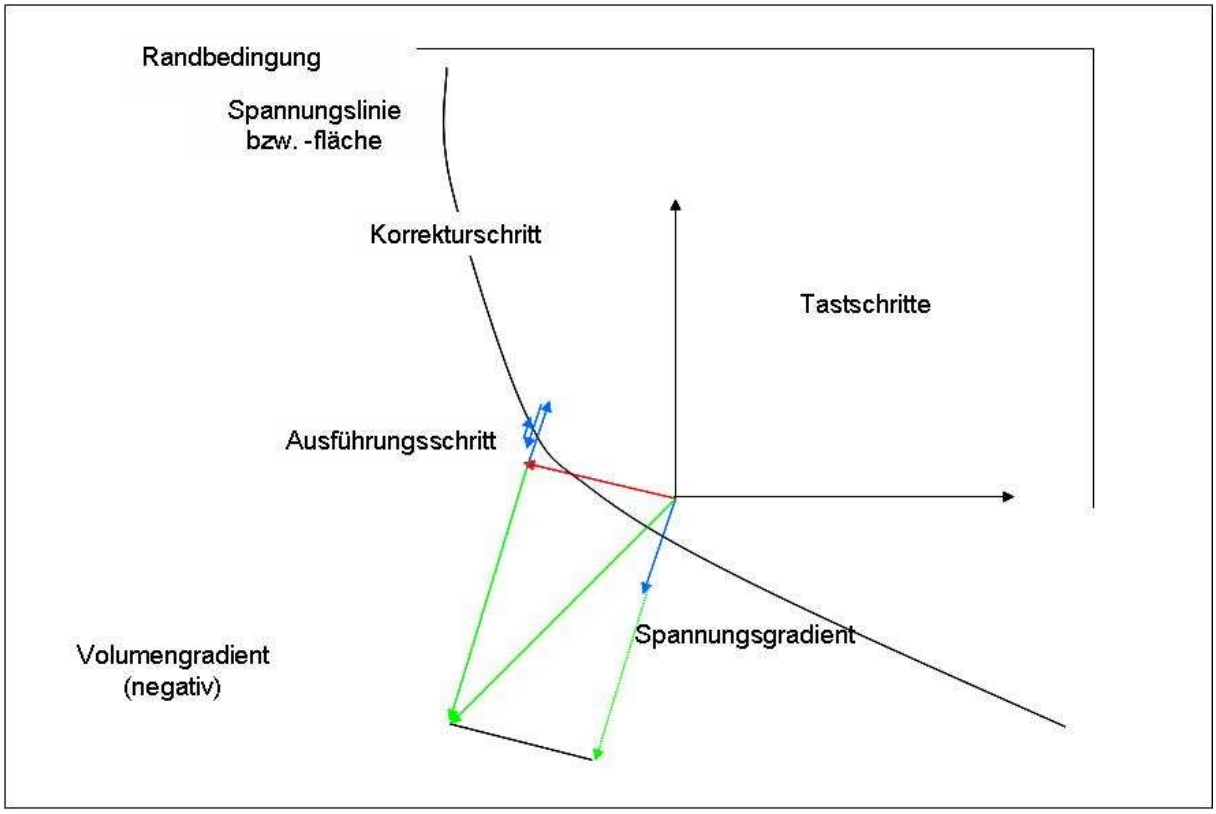


Abb. 3.5.4: Skizze des Algorithmus Vektorprojektion-Sigmakorrektur

Aus Gründen, die im folgenden Kapitel erörtert werden, gibt es diesen Algorithmus in drei Variationen. Dabei betreffen die Unterschiede die Datenaufnahme bzw. die Sensitivitätsanalyse. Der schon genannte Algorithmus Vektorprojektion-Sigmakorrektur halbiert, wie schon erwähnt, die Schrittweiten auch schon bei den Tastschritten. Dann gibt es den Algorithmus *Vektorprojektion-Sigmakorrektur-TastConst*. Dieser behält bei der Sensitivitätsanalyse die Schrittweite bei, arbeitet bei den Ausführungsschritten mit Variablen Schrittweiten. Weiterhin gibt es den Algorithmus *Vektorprojektion-Sigmakorrektur-Mittelwert.* Dieser benutzt zur Berechnung des Spannungsgradienten nicht wie alle anderen Algorithmen die maximalen Spannungswerte, sondern den Mittelwert aus den Spannungen aller einzelnen Elemente des Bauteils.

3.6. Der Algorithmus „Penalty“

Eine andere Strategie verfolgt der Algorithmus *Penalty-Gradient-Linear*. Er nutzt nicht direkt die Sigmalinie zur Ausrichtung des Ausführungsschrittes und führt keine Abfrage durch, ob er sich im zulässigen Raum befindet, um ihn gegebenenfalls zu verlassen. Penalty-Gradient-Linear ist von seinem Ausführungsschritt ein reines *Line-Search-Verfahren*, d.h. er führt einfach eine Gradientenermittlung durch und geht dann in Richtung des steilsten Abstiegs, ohne auf die Restriktion zu achten. Die Restriktion bezieht er allerdings in die Sensitivitätsanalyse mit ein. Und zwar insofern, als dass der nichtzulässige Raum einen gewissen Ergebniswertzuschlag, also eine künstliche Vergrößerung des Volumens erhält. Auf diesem Wege wird er bewusst wieder aus dem unzulässigen Raum herausgeleitet. Je nach dem Abstand zur Sigmafläche, ist die entstehende rücktreibende Kraft unterschiedlich groß.

In diesem Zusammenhang ist es existentiell wichtig, den Volumenaufschlag richtig zu wählen, um zu gewährleisten, das der Algorithmus zum einen den Weg zurück in die zulässige Zone findet und zum anderen, das die rücktreibende Kraft nicht zu groß ist, und dadurch die Parallelbewegung zur Sigmalinie vollständig überlagert wird. Aus diesem Grund muss der Aufschlag über ein Potential genau gesteuert werden. Dieses Potential muss demnach laufend neu ermittelt werden (**Abbildung 3.6.2**).

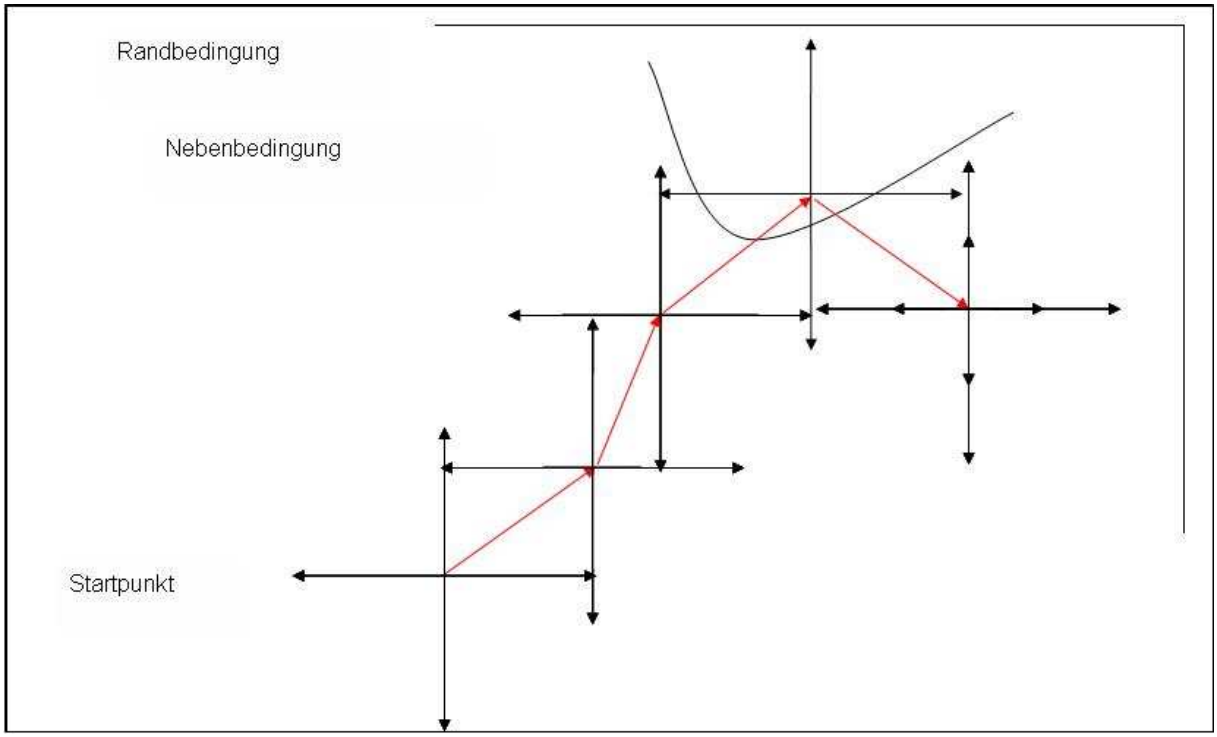


Abb. 3.6.1: Skizze des Algorithmus Penalty-Gradient-Linear

$$\phi_1 = \frac{\Delta V_1}{\Delta \sigma_1}$$
$$\phi_2 = \frac{\Delta V_2}{\Delta \sigma_2}$$

Gl. 3.6

Das Potential Φ_i ist die Änderung des Volumens pro der Änderung der Spannung in der jeweiligen Variablenrichtung. Die Erfassung dieser Werte läuft parallel zur Sensitivitätsanalyse, also während des Tastschrittzyklus.

$$\vec{v} = \begin{pmatrix} \Delta V_1 \\ \Delta V_2 \end{pmatrix} = \begin{pmatrix} \Delta V_1 / \Delta V_2 \\ \Delta V_2 / \Delta V_2 \end{pmatrix} = \begin{pmatrix} a \\ 1 \end{pmatrix}$$

Gl. 3.7

Die größere Volumenänderung erhält den Proportionalitätsfaktor 1, die anderen Variablen erhalten die entsprechende Normierungsgröße. Mit diesen Informationen wird in Gl. 3.7 das Gesamtpotential berechnet.

$$\phi = \phi_1 * a + \phi_2 * 1$$

Gl. 3.8

In **Abbildung 3.6.2** ist zu sehen, wie der Potenzialwert sinnvoll in dem Algorithmus umgesetzt wird. Angenommen, ein Ausführungsschritt zeigt von dem Punkt 0 zu dem Punkt 1', dann sind die Volumen und Spannungswerte an diesem Punkt nach durchgeführter Berechnung bekannt. Indem die Spannungswerte subtrahiert werden, entsteht Δσ. Dieser Wert mit dem ermittelten Potential multipliziert und auf das Volumen V_1' zugeschlagen, ergibt je nach verwendetem Faktor die in **Abbildung 3.6.2** dargestellten Kurvenverläufe. In diesem Programm wurde die Spiegelung der Kurve gewählt, also der Faktor 2. Dass die Kurve einen Knick aufweist und damit unstetig ist, erschien nicht als Problem, da bei der Gradientenbildung nicht mit den Ableitungen gerechnet wird, sondern diese über das Differenzenverfahren angenähert werden.

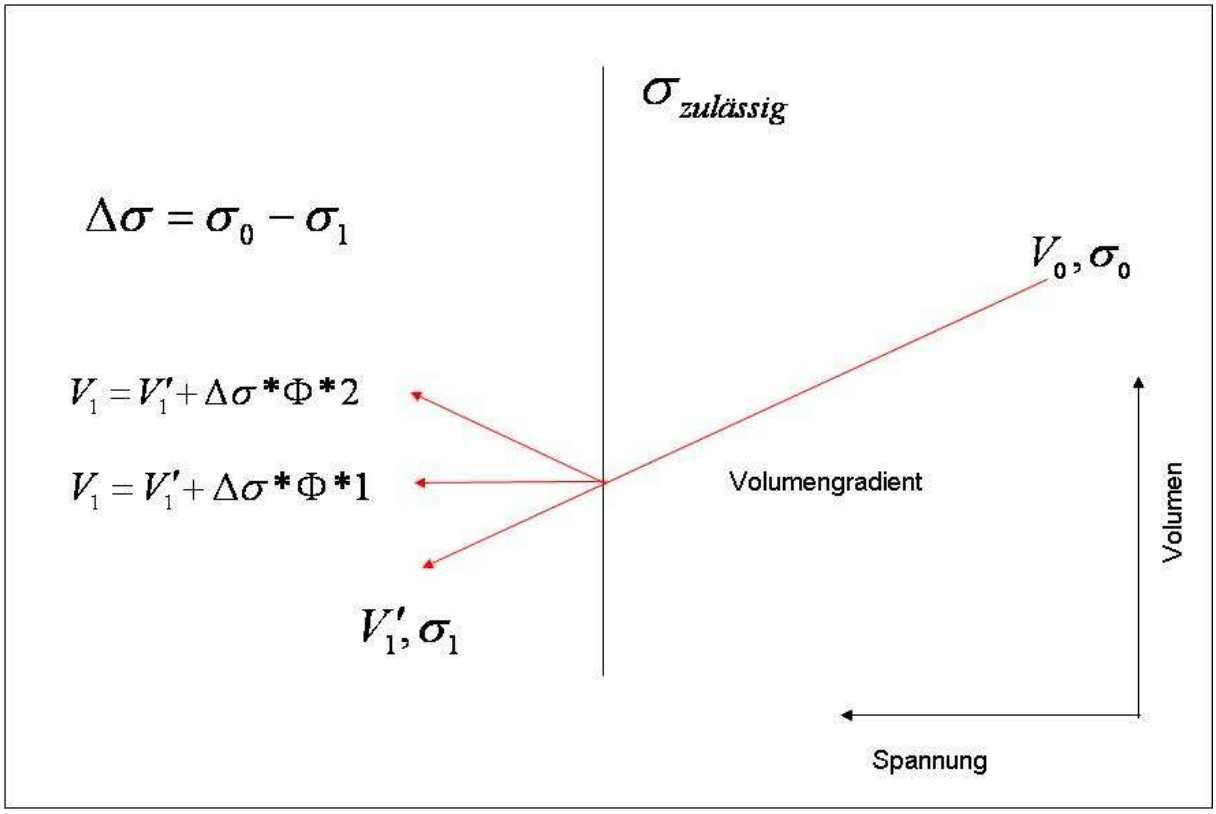


Abb. 3.6.2: Potentialzuschlag

Auch den Algorithmus Penalty gibt es in drei Ausführungen. Diese werden in den folgenden Skizzen dargestellt.

In **Abbildung 3.6.3** ist der bisher beschriebene Penalty-Gradient-Linear zu sehen. Ein Schritt ist zu sehen, in dem der Algorithmus den unzulässigen Bereich wieder verlässt. In **Abbildung 3.6.4** ist eine Neuerung eingeführt worden. Es kann geschehen, dass eine von den hier dargestellten zwei Variablen einen sehr dominierenden Volumengradienten hat. In diesem Fall ist es sehr wahrscheinlich, dass der sich ergebende Gesamtgradient zu einem sehr großen Teil in die Richtung der einen Variablen zeigt, so dass die resultierenden Iterationsschritte weitestgehend

hin- und herpendeln und entlang der Sigmalinie, in Richtung der zweiten Variablen, kaum Raum gutgemacht wird. In so einem Fall kann es helfen, die Steigung der Spannungslinie zu ermitteln, und wie dargestellt dem Gradienten überlagernd zuzurechnen.

Als dritte Möglichkeit wurde die Zusammenführung der beiden Gruppen Penalty und Vektorprojektion entwickelt. Hier wird zum einen die Ermittlung der Gradienten Volumen und Spannung und die anschließende Vektorprojektion durchgeführt, gleichzeitig aber auch die Potentialermittlung und der Volumenaufschlag **(Abbildung 3.6.5)**.

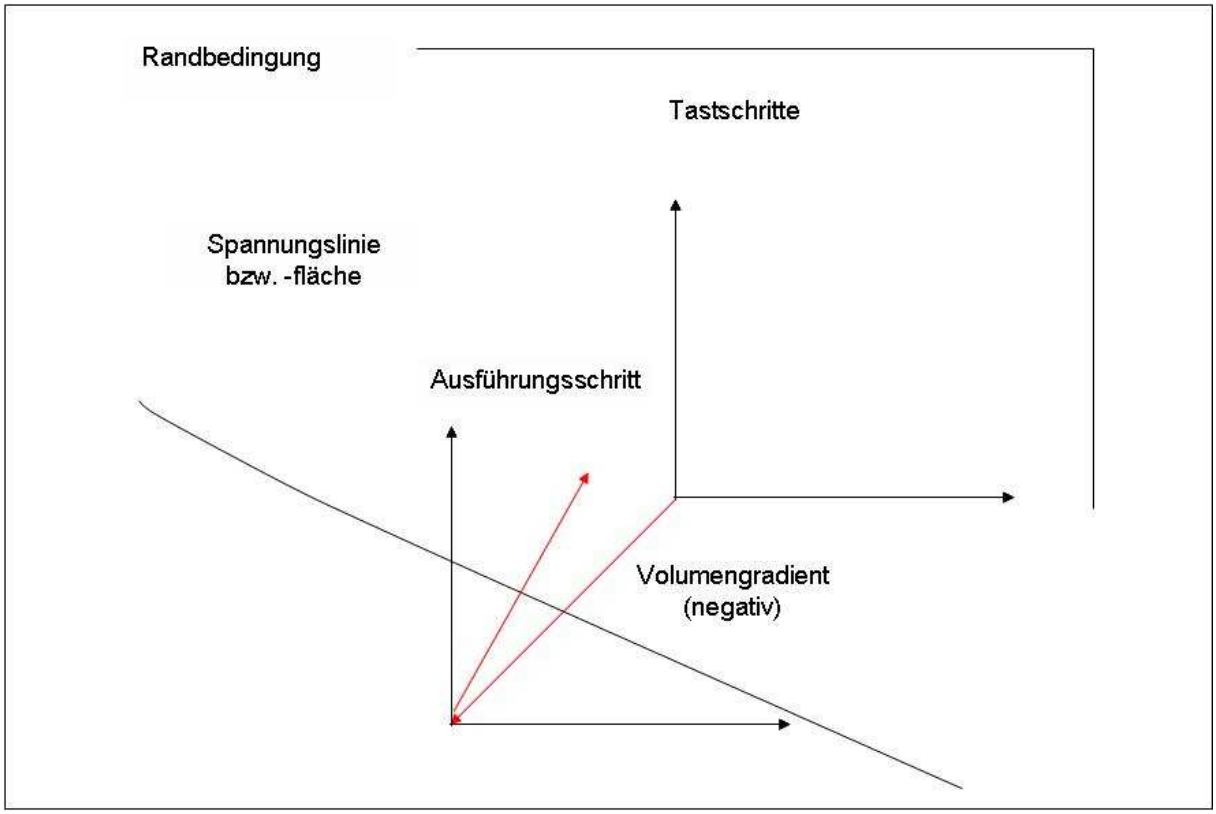


Abb. 3.6.3: Skizze des Algorithmus Penalty-Gradient-Linear

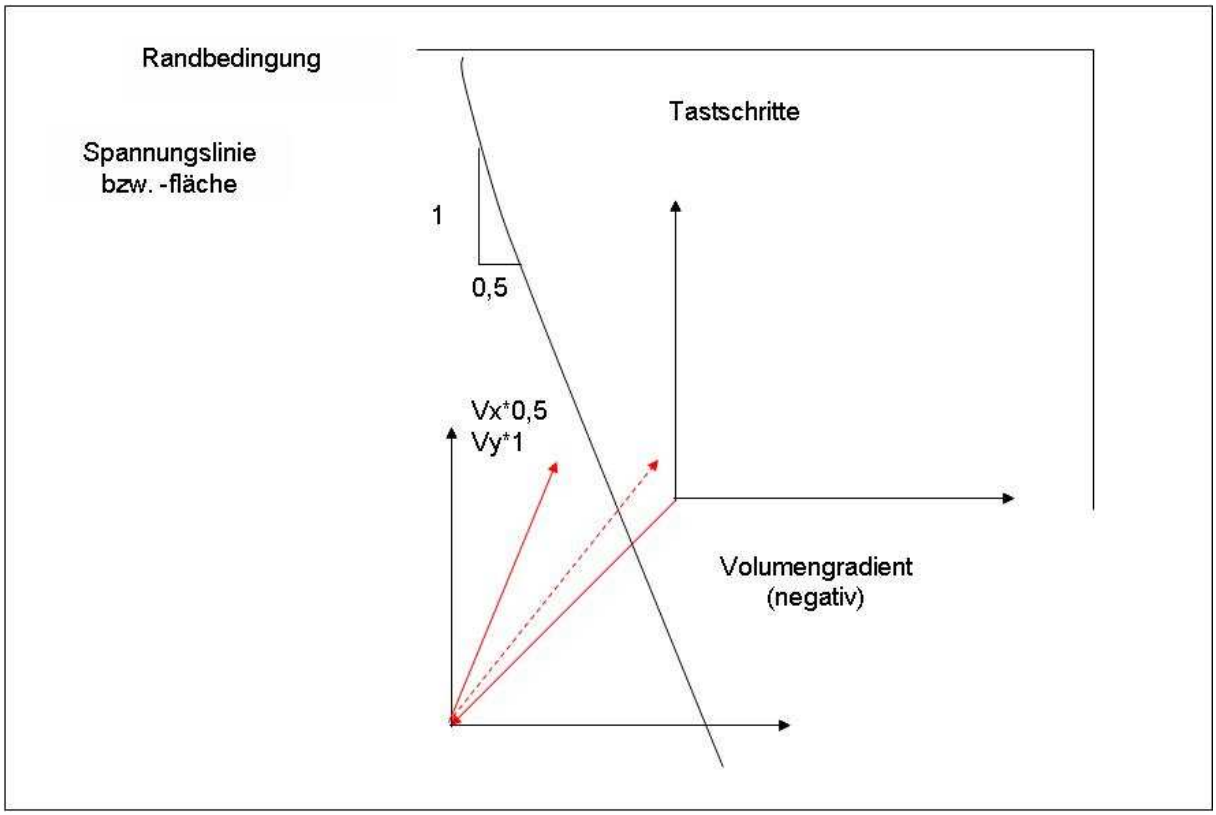


Abb. 3.6.4: Skizze des Algorithmus Penalty-Spannungsgewichtet

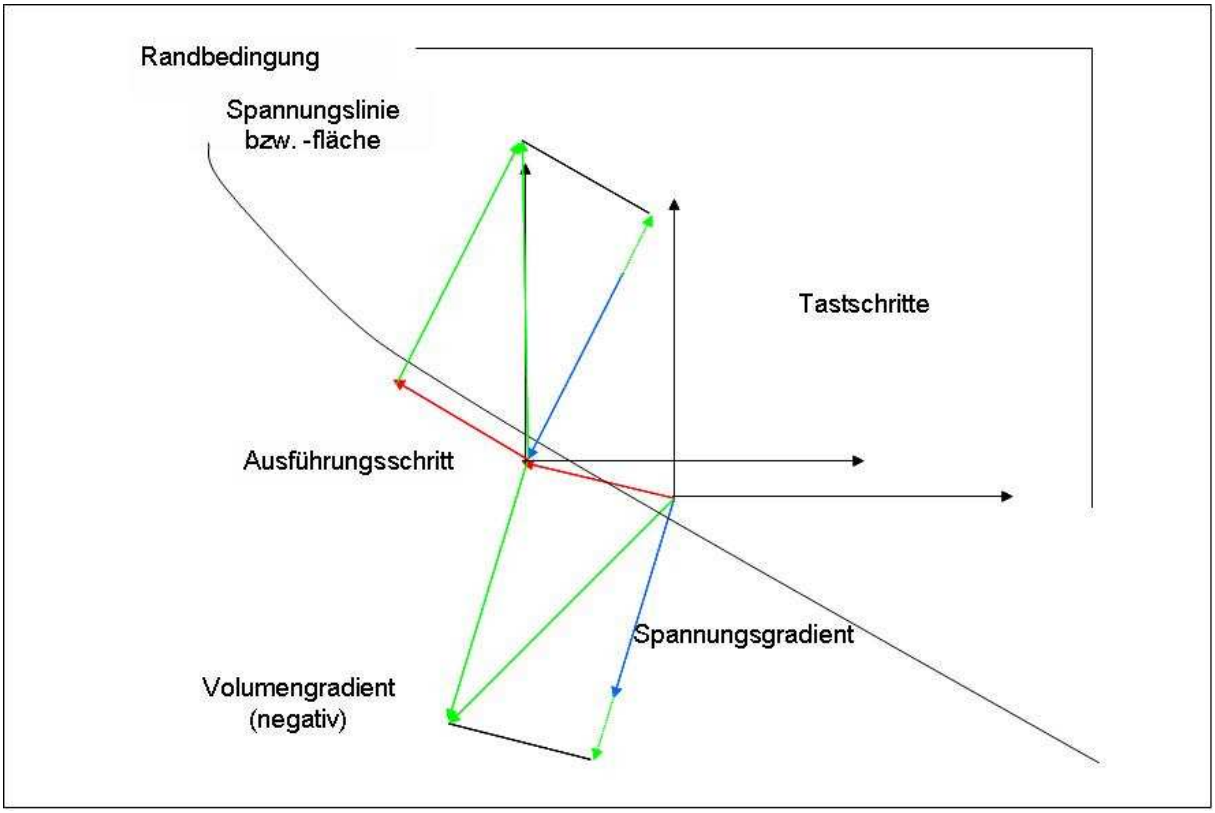


Abb. 3.6.5: Skizze des Algorithmus Penalty-Vektorprojektion

3.7. Der Algorithmus „Plus-Minus“

Der Algorithmus Plus-Minus überprüft sämtliche mögliche Kombinationen der Änderungen der Variablen. Bei drei Variablen sind das die Eckpunkte eines Quaders. Vom Mittelpunkt des Quaders ausgehend hat man die Möglichkeit z.B. alle Variablen gleichzeitig zu vergrößern (+++) oder alle zu verkleinern (---). Um hier alle Möglichkeiten programmieren zu können, braucht man folgenden Algorithmus:

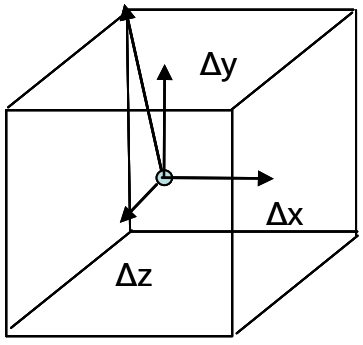
Anzahl der Kombinationen: 2^V

V = Anzahl der Variablen

Für drei Variablen: $2^3 = 8$

Nun werden die Plus und Minus in der folgenden Weise angeordnet **(Abbildung 3.7.1).** Bei 8 Zeilen wird in die Zeilen der ersten Hälfte + geschrieben und in die der unteren Hälfte -. In der zweiten Spalte werden + und – in der gleichen Weise angeordnet, nur in einem nochmals halbierten Intervall. Das gleiche gilt für die dritte Spalte. Nun lassen sich zeilenweise die Kombinationsmöglichkeiten ablesen.

Der hier dargestellte Hilfsalgorithmus lässt sich programmtechnisch umsetzen, sodass sämtliche Kombinationsmöglichkeiten abgearbeitet werden können.



Δx	Δy	Δz
+	+	+
+	+	-
+	-	+
+	-	-
-	+	+
-	+	-
-	-	+
-	-	-

Abb. 3.7.1: Kombinationsmöglichkeiten im 3-Dimensionalen

Der Algorithmus Plus-Minus könnte im zweidimensionalen Fall den unten dargestellten Verlauf annehmen. Die vier Ecken des Quadrats werden untersucht, und wenn eine Verbesserung eingetreten ist, wird der beste Wert übernommen. Dies funktioniert aber nur wenn die Schrittlängenverhältnisse eine Verbesserung bei Einhaltung der Restriktion zulassen, wenn also eine Hauptrichtung der Variablenänderung in etwa parallel zur Sigmalinie ist.

Um dieses Problem behandeln zu können, untersucht der Algorithmus nicht nur die Ecken des dreidimensionalen Würfels oder des n-dimensionalen Polyeders, sondern er geht auch die Seitenkanten des Problems ab, hat also dadurch einen Variablen und nicht einen festen Winkel **(Abbildung 3.7.3).**

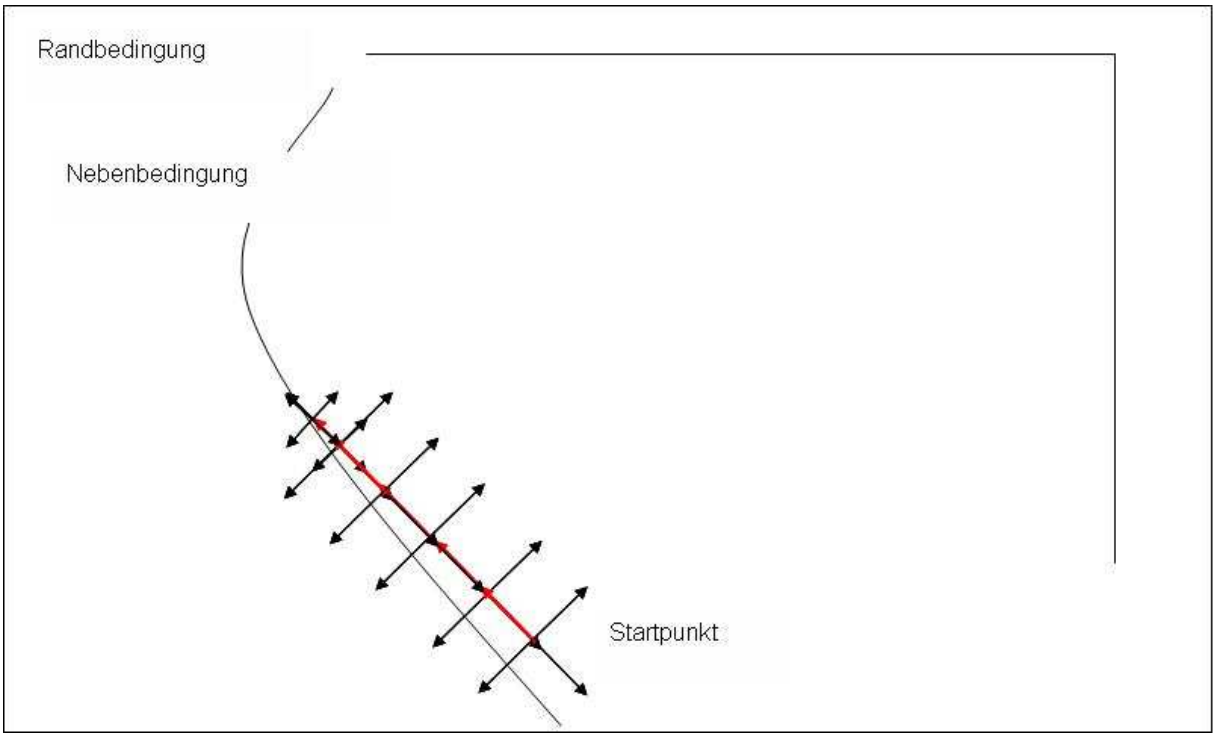


Abb. 3.7.2: Skizze des Algorithmus Plus-Minus

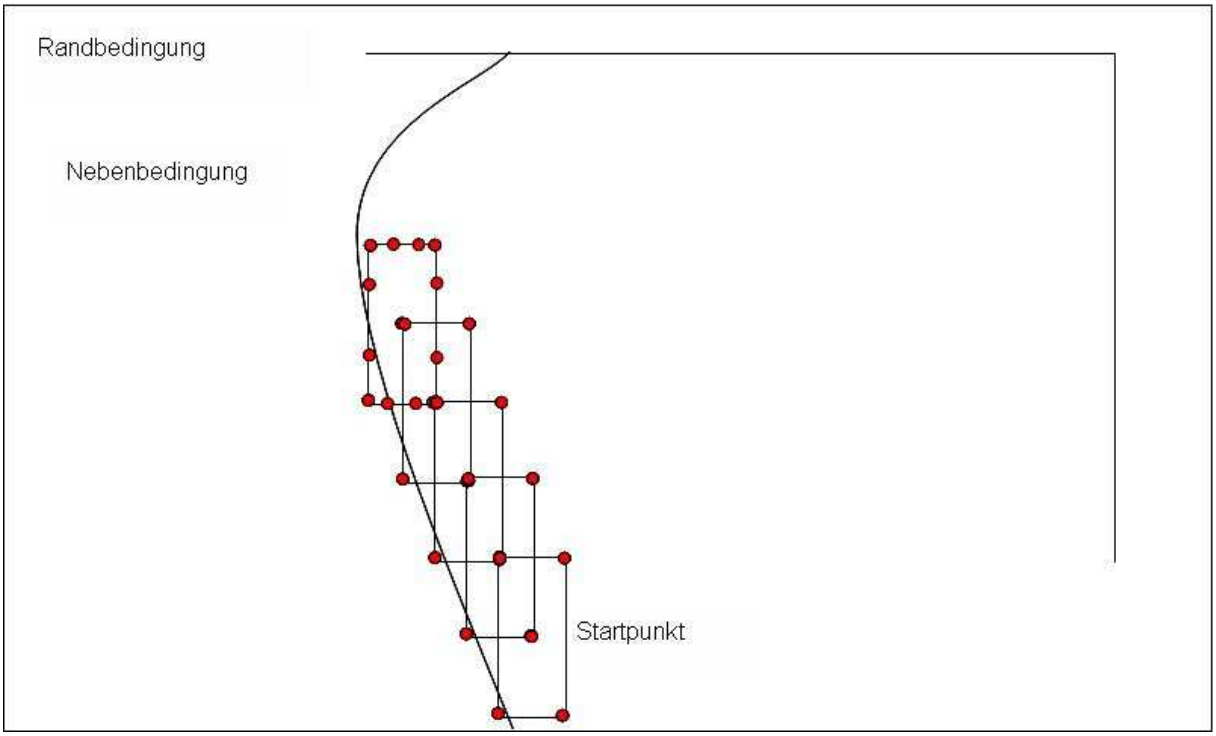


Abb. 3.7.3: Skizze des variablen Algorithmus Plus-Minus

3.8. Der Algorithmus „Plus-Minus-Global“

Wie der Name schon sagt, handelt es sich bei diesem Algorithmus um eine Variation des Algorithmus Plus-Minus, der prinzipiell die Möglichkeit hat, ein lokales Minimum zu verlassen und weiter zu suchen und so ein besseres, vielleicht das globale Minimum zu finden **(Abbildung 3.8.1)**.

Dies ist so vorsichtig formuliert, da man sich nie sicher sein kann, das globale Minimum gefunden zu haben, oder es sein kann, dass, wie immer bei globalen Algorithmen, der Aufwand zu groß wird, und die Suche deshalb einfach abgebrochen wird.

Die Funktionsweise ist die, dass nachdem ein Minimum gefunden wurde und die vorherige Rechnung beendet wurde, von diesem Minimum der globale Optimierer gestartet wird. Dieser vergrößert nun die Längen der Tastschrittweiten, in der Annahme, dass mit diesen hinter der „Erhebung“, die das lokale Minimum umgibt, ein niedrigerer Wert gefunden werden kann.

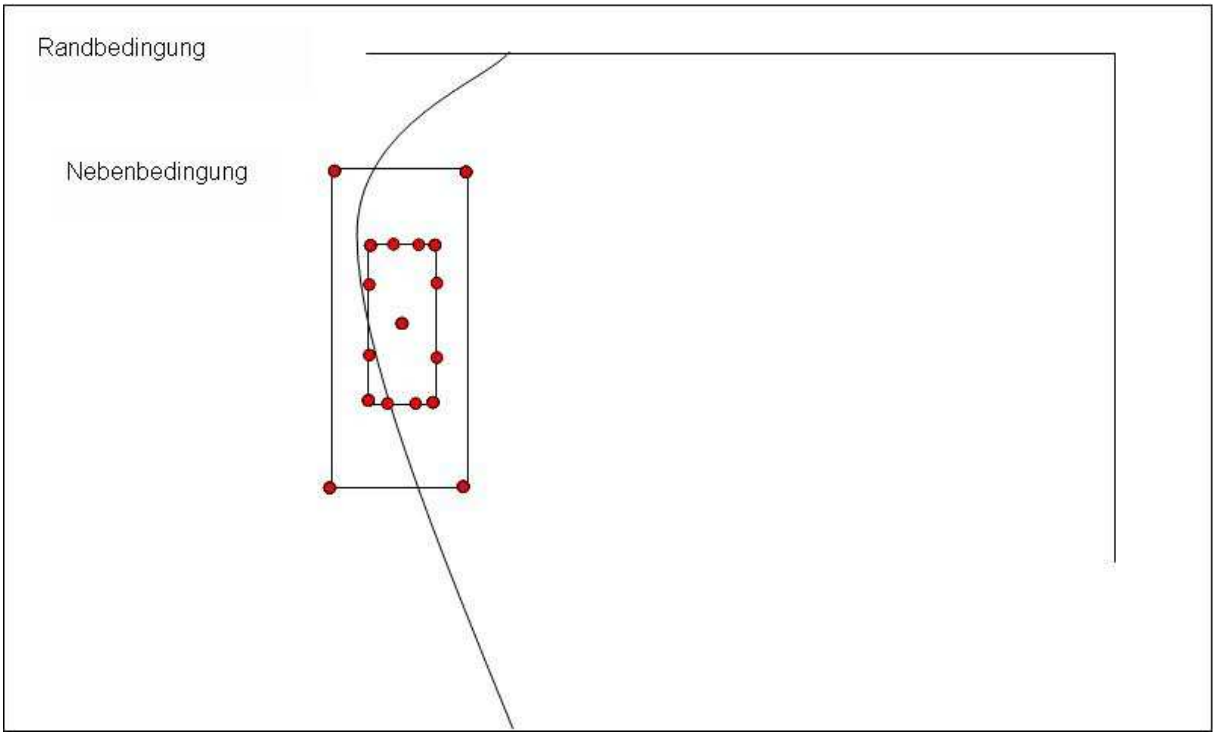


Abb. 3.8.1: Skizze des Algorithmus Plus-Minus-Global

3.9. Verifizierung des Beispiels 3 Stäbe aus Kapitel 1

Am Beispiel 3 Stäbe aus Kapitel 1 sollten die erstellten Algorithmen überprüft werden. Das Beispiel wurde mit einem speziellen FORTRAN-Programm optimiert. So sind die Ergebnisse schon bekannt, mit denen die neuen Ergebnisse verglichen werden können. Um diese neuen Ergebnisse zu erhalten, wurde ein Stabmodell für Abaqus entworfen, das mit dem Optimierungsprogramm bearbeitet werden kann. Zum Vergleich wurden die Algorithmen Plus-Minus und Vektorprojektion-Sigmakorrektur herangezogen. Die Ergebnisse sind in **Abbildung 3.9.1 und 3.9.2** dargestellt. Die angegebenen Variablen A_i sind die Stabquerschnitte in mm².

Alpha	A_1 [mm²]	A_2 [mm²]	A_3 [mm²]
2,5	0,1	444	319
1,5	509	0,1	0,1
1	524	0,1	0,1

Abb.3.9.1: Ergebnisse von Plus-Minus

Alpha	A_1 [mm²]	A_2 [mm²]	A_3 [mm²]
2,5	69,5	345	234
1,5	505	11	4
3,5*	0,1	430	203

Abb.3.9.2: Ergebnisse von Vektorprojektion-Sigmakorrektur

Die Ergebnisse der Rechnungen mit Plus-Minus zeigen eine klare Übereinstimmung mit den Referenzrechnungen. Die errechnete Querschnittsfläche von 0,1 mm² ist die untere angegebene Grenze. Im Verhältnis zu den anderen berechneten Flächen ist 0,1 vernachlässigbar. Zu den Ergebnissen der Rechnungen mit Vektorprojektion-Sigmakorrektur ist festzustellen, dass die Rechnung mit α=1,5 eine recht gute Annährung an das Optimum erzielt, während α=2,5 nur in der Tendenz richtig liegt. Das liegt daran, dass bei manchen Gradientenermittlungsschritten die Spannung bei Änderung der Variablen 1 keine Änderung erfährt, wodurch für diese Variable kein oder ein sehr geringer Anteil am Ausführungsvektor ermittelt wird. Dieser Effekt wird *entkoppelte Variable* genannt und wird in den folgenden Kapiteln eingehender behandelt. Für dieses Beispiel lässt sich dieser Effekt verhindern, indem die angreifende Kraft nicht mehr parallel zu Stab eins verläuft. Führt man eine solche Änderung durch, wird die Variable 1 auch mit dem Algorithmus Vektorprojektion-Sigmakorrektur optimiert und erreicht den Wert 0,1 mm² (Abb.3.9.2: 3,5*).

4. Anwendung der Algorithmen

In den vorherigen Kapiteln wurde der prinzipielle Aufbau der Optimierung geschildert und die Funktionsweise bzw. der Aufbau der entwickelten Algorithmen. In diesem Kapitel wird die Optimierung beispielhaft angewendet werden. Die Einsatzfähigkeit wird an zehn Beispielen überprüft werden. Diese Beispiele sind zehn ABAQUS/CAE-Modelle, die als eigenständige Programme lauffähig, rechenbar sind. Sie sind aber schon soweit weiterentwickelt, dass sie auch zur Optimierung taugen, d.h. dass sie so konzipiert sind, dass sie auch noch nach einer, vom Algorithmus durchgeführten Änderung der Variablenwerte, wiederhergestellt werden können und gerechnet werden können. Dies ist nicht selbstverständlich, sondern ist die Aufgabe des Anwenders, nämlich ein optimierungslauffähiges Modell zu entwerfen. Weiterhin muss das Modell auch zu optimieren sein, also ein Optimierungsproblem sein bzw. ein Optimum haben, da sonst der Algorithmus bestenfalls direkt an die Systemgrenzen läuft. Die ersten neun Optimierungsbeispiele sind Gewichtsoptimierungen mit einer Spannungsrestriktion und erst das letzte Beispiel ist eine restriktionsfreie Optimierung. Bei dieser soll die Steifigkeit einer Struktur erhöht werden.

Die Modelle werden nacheinander, in Unterkapiteln strukturiert, behandelt werden. Zuerst wird eine kurze Beschreibung des FEM-Modells, seine Belastungen und Randbedingungen und des aus ihm entstehenden Optimierungsproblems erfolgen. Danach wird die Durchführung der Optimierung beschrieben, in welcher Reihenfolge welche Algorithmen genutzt wurden, mit welchen Startwerten gerechnet wurde, welcher Rechengang welches Ergebnis gebracht hat. Dies ist also die Darstellung der jeweiligen Optimierungsstrategie. Dazu gehört die Darstellung der Ergebnisse, die die verwendeten Algorithmen gebracht haben.

Im letzten Unterkapitel werden die Ergebnisse ausgewertet und es wird versucht, prinzipielle Unterschiede zwischen den einzelnen Modellen herauszuarbeiten und darzustellen. Dabei soll dargestellt werden, welcher Algorithmus für welches Problem am besten geeignet ist

4.1 Das Modell „2 Stäbe“

4.1.1 Der Modellaufbau

Das Modell ist ein Stabtragwerk, das aus zwei Stäben besteht. Die linken Stabenden sind in ihren Verschiebungsfreiheitsgraden fest gelagert und das rechte Stabwerkende ist durch eine Kraft F belastet. Das Modell „2 Stäbe“ (**Abbildung 4.1.1**) wird mit einem linear-statischen Code gerechnet. Dies betrifft nicht den Rechenlauf der Optimierung, sondern bedeutet, dass jede einzelne in Abaqus definierte Rechnung der Optimierung linear-statisch ist. Die Geometrie wurde als Stab bzw. Strebe (Truss) erstellt. Die Vernetzung besteht aus zwei Elementen (Truss). Truss-Elemente können nur axiale Kräfte aufnehmen und keine Schubkräfte oder Momente. So kann mit ihnen ein Stabtragwerk erstellt werden. Der untere Stab hat die während der Optimierung fixe Länge von 1000 mm. In den Abmaßen der Abbildung hat die Variable 1 den Wert 850 mm und die Variable 2 den Wert 130 mm². Die Variable 2 ist die Schnittfläche A der Stäbe. Die Kraft F teilt sich bei diesem Beispiel in die Anteile F_x = 100 KN und F_y = 30 KN.

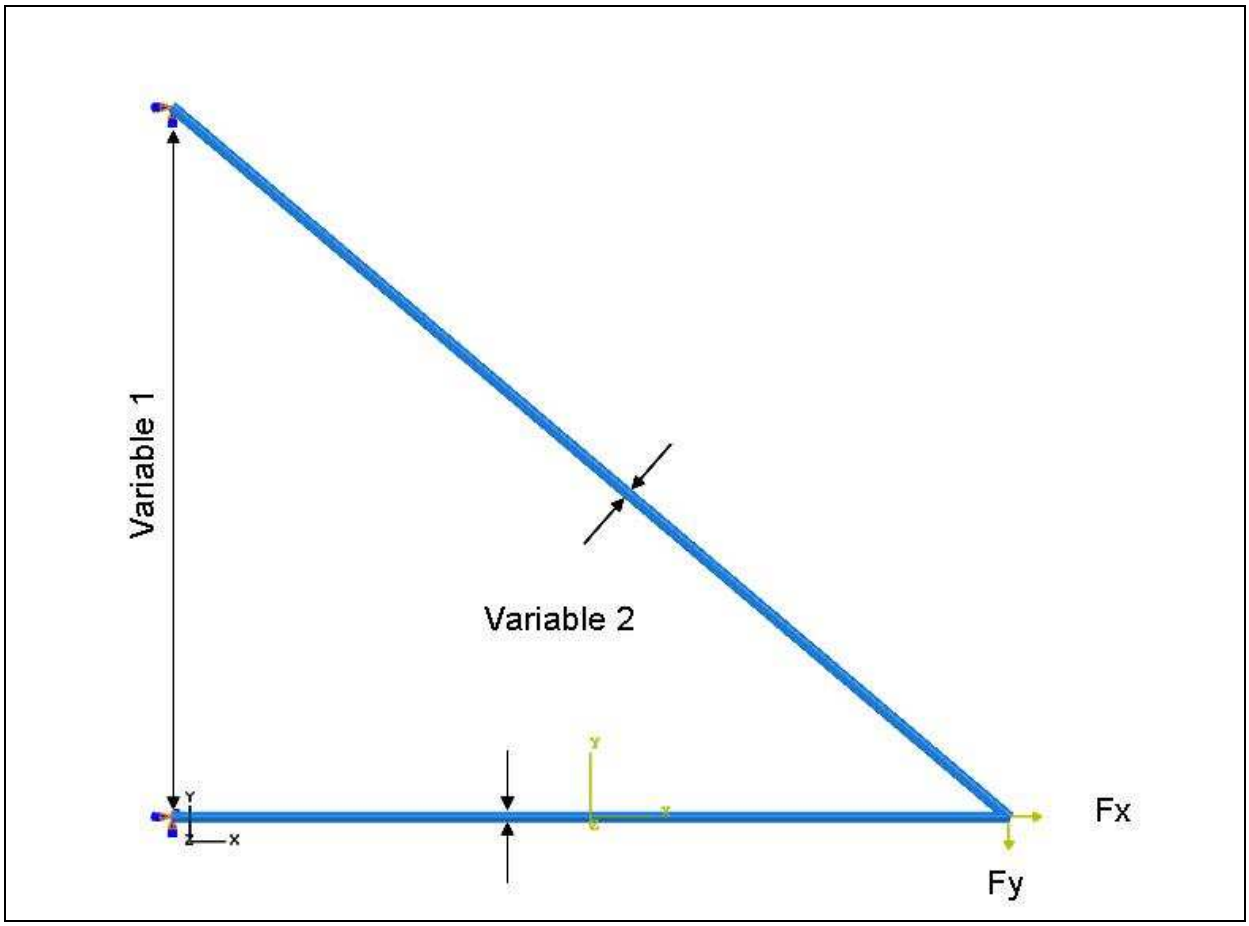


Abb. 4.1.1: Das Modell 2 Stäbe

Die Variable 1 wird im Sketcher-Modul von Abaqus/CAE definiert. Im Sketcher wird die Geometrie des Modells erstellt oder bearbeitet. Im Laufe der Lernphase wird der

Sketcher geöffnet und die in **Abbildung 4.1.2** gezeigte Länge durch *Add Dimension Value* (Markierung) definiert. Dies kann auch schon vor der Lernphase, also vor dem eigentlichen Beginn der Optimierung geschehen und gespeichert werden. In der Lernphase muss jedoch die Änderung des Dimension Values durchgeführt und so die Variable definiert werden (*Edit Dimension Value*; Markierung). Wie schon erwähnt, benötigt die eingegebene Zahl den Zusatz 0001, z. B. 500,0001. Dies ist zur Identifizierung notwendig. Der Sketcher kann in dem Modul *Part* aufgerufen werden. Die Definition der zweiten Variablen wird im Modul *Eigenschaften* (Property) durchgeführt. Dort ist in dem Fenster *Section/ Section Manager/ Section Edit* der Wert *Cross Sectional Area* mit dem Zusatz 0001 zu versehen, um so die Variable zu markieren und dadurch zu definieren.

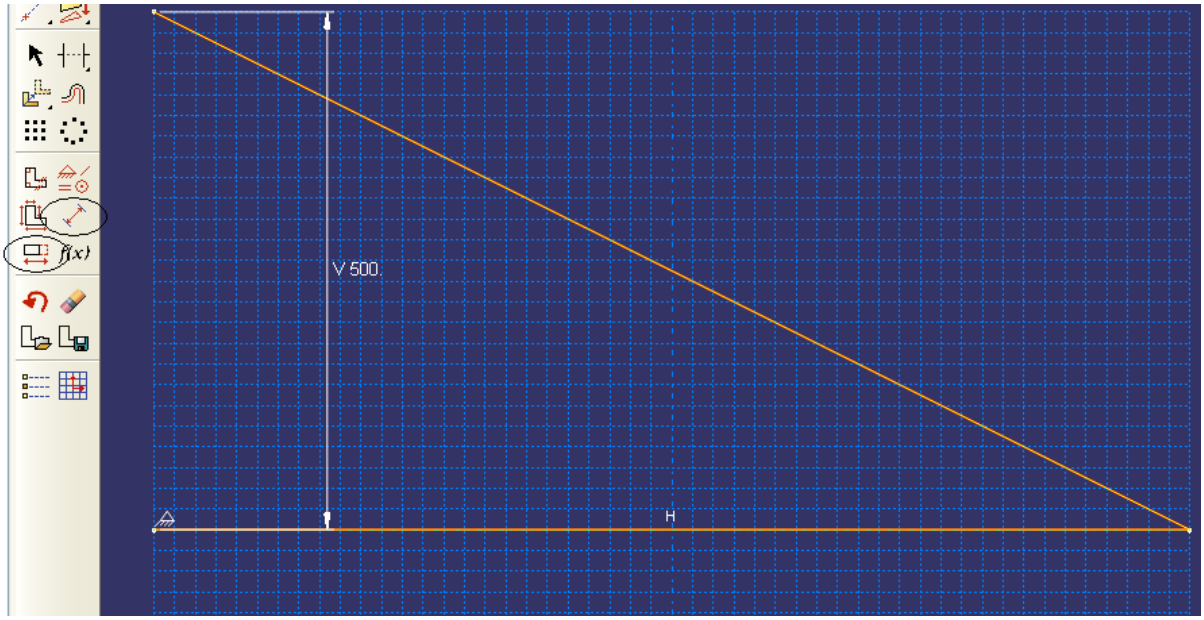


Abb. 4.1.2: Sketcher

Das Modell ist ein Optimierungsproblem mit einer Restriktion. Es soll gewichtsoptimiert werden bei Einhalten der zulässigen Spannung. Die Zielfunktion ist demnach das Gewicht bzw. das Volumen der beiden Stäbe. Deshalb müssen sie in einem Zielwertset im Modell als eigenes Abaqus-Set gespeichert werden. Der Name dieses Sets ist frei wählbar, denn er wird im Fenster der Optimierung eingegeben. In dieser Arbeit erhalten die Elementzielwertsets alle den Namen *ELEMENTE*. Der Anwender hat dafür zu sorgen, dass die Misesspannungen und das Elementvolumen in die Ausgabedatei geschrieben werden.

Ziel der Optimierung ist die Minimierung des Volumens. Das bedeutet, dass beide Stäbe möglichst nahe an der Spannungsgrenze sein sollen. Eine vorherige Abschätzung des Ergebnisses ist sehr schwierig, da irgendwo im Feld die größte Belastung eines Stabes auf den anderen Stab umschlägt. Bei welchem Winkel zwischen den Stäben und zwischen der Belastung die Stäbe nun weitestgehend

dünn gebaut werden können, ist schwer abzuschätzen, obwohl nur zwei Variablen vorhanden sind.

4.1.2 Der Algorithmus Gradient-Zurück

Der Vorteil an zweidimensionalen Systemen ist, dass die Ergebnisse recht anschaulich in Diagrammen darzustellen sind. Die Bewegung des Algorithmus durch das Feld ist zu sehen und die Sigmalinie ist eingezeichnet. Das Volumenfeld selber ist allerdings nicht zu sehen. Dies käme senkrecht aus der Blattebene heraus. Davon bekommt man allerdings einen Eindruck, da einige Werte des Volumens eingetragen wurden. Der Volumengradient der Querschnittsfläche der Stäbe ist größer als der der Variable 1 (Länge). Der Gesamtspannungsgradient steht senkrecht auf der Sigmalinie. Dies bedeutet, dass bei kleineren Winkeln zwischen den Stäben der Einfluss des Winkels auf die Spannung überwiegt, während bei größeren Winkeln der Einfluss der Querschnittsfläche die Überhand gewinnt. Nach den ersten Rechnungen ist der beste Wert in dem Bereich von 150 mm / 550 mm² zu suchen. Das Wertepaar richtet sich nach dem Diagramm und nicht nach der Variablennummer, das heißt, zuerst wird der Abszissenwert genannt.

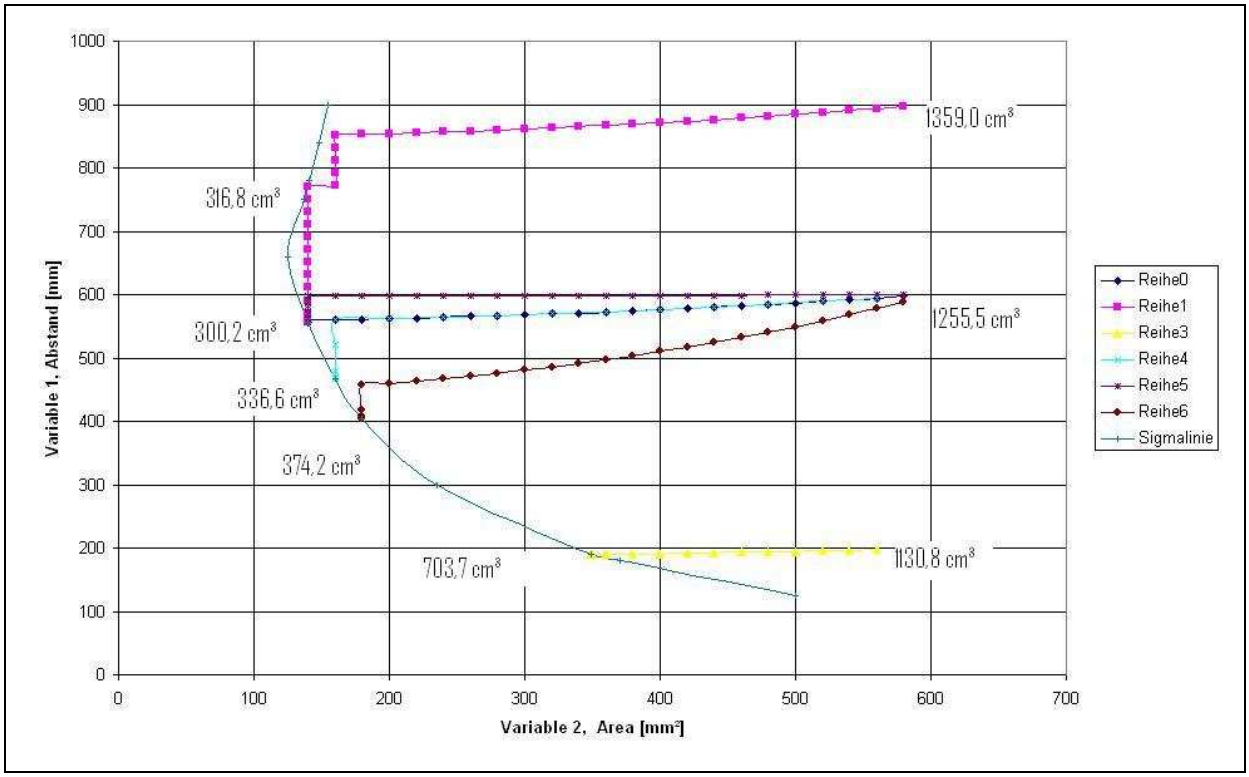


Abb. 4.1.3: Diagramm des Modells 2 Stäbe (Algorithmus Gradient-Zurück)

Die ersten Rechnungen sind die Ergebnisse des Algorithmus Gradient-Zurück (**Abbildung 4.1.3**, Reihe0 bis Reihe6). Die unterschiedlichen Ergebnisse bei unterschiedlichen Startwerten sind zu ersehen. Die Rechnungen starten bei einem

Querschnitt von 600 mm² und unterschiedlichen Abständen von 200 mm bis 900 mm. Die dargestellten Werte sind die Ausführungsschritte, also die ermittelten Gradienten. Zu deren Ermittlung wurden die Variablen in vorhergehenden Tastschritten variiert, indem von dem jeweils ersichtlichen Punkt die Schrittweite addiert und subtrahiert worden ist. Um jeden dieser Punkte gehören also vier Tastschritte, die der Übersichtlichkeit wegen, nicht dargestellt sind. Es sind einige dutzend Ausführungsschritte für jeden Optimierungslauf, bis die Abbruchbedingung greift. Bis dahin überwiegt der Anteil des Querschnitts am Gradienten, sodass in erster Linie dieser Anteil verringert wird. Wenn ein Tastschritt die Spannung verletzt, wird dieser aus der Gradientenermittlung herausgenommen. In diesem Fall laufen dann die Gradienten sehr idealtypisch querab zur Sigmalinie. Dies geschieht jedoch nur, wenn ein Gradientenanteil dominiert. Weiter ist hier zu erkennen, dass dieser Algorithmus einzelne Variablen nicht verschlechtern kann, das bedeutet, dass wenn ein Tastschritt einer Variablenrichtung das Ergebnis verschlechtert, geht der Ausführungsschritt nicht in diese Richtung. So kann der Algorithmus in diesem Fall nur nach links und nach unten gehen. Das bedeutet, dass er zwar nicht fälschlich sich vom Optimum entfernt, jedoch kann er, wie bei Reihe 3 der Abbildung 4.1.3 zu sehen, verfrüht an der Sigmalinie zum Stillstand kommen.

Die Reihen null, vier, fünf und sechs starten mit den gleichen Startwerten, trotzdem verlaufen ihre Wege teilweise unterschiedlich und sie erreichen nicht immer die gleichen Endwerte. Dies liegt an der Konzeption des Algorithmus Gradient-Zurück, die schon im vorherigen Kapitel beschrieben wurde. Danach wird der Ausführungsschritt, der Gradient, erstellt, indem die Tastschritte mit ihren Schrittlängen je nach der Änderung des Ergebniswertes, den sie erwirkten, gewichtet werden. Dies hat zur Folge, dass ein größerer Tastschritt meistens eine größere Änderung bewirkt, und dadurch eine höhere Gewichtung in der Vektorerstellung bekommt. So bewirken unterschiedliche Verhältnisse zwischen den Tastschrittlängen, unterschiedliche Tendenzen in der Ausrichtung der Wege der Algorithmenrechnungen (Reihe null, fünf und sechs). Bleibt das Verhältnis konstant und beide Tastschrittlängen werden gleich geändert, kann, wie in diesem Feld, die Richtung des Weges gleich bleiben (Reihe null und vier). Erst an der Sigmalinie macht sich die größere Schrittlänge von Reihe vier bemerkbar. Sie ist eher bei einem Tastschritt im nicht zulässigen Bereich und ändert entsprechen die Richtung. Durch diese Art der Gradientenerstellung hat der Anwender die Möglichkeit, auf die Richtung einen gewissen Einfluss zu nehmen.

4.1.3 Penalty

Der Algorithmus Penalty ist in drei Ausführungen vorhanden. Diese werden für das Modell 2 Stäbe alle eingesetzt. Den Anfang macht Penalty-Gradient-Linear. Dieser ist, wie Gradient-Zurück vor eine Bewegung durch das Feld auch losgelöst von der

Sigmalinie einsetzbar. Die ist für die Algorithmen Penalty-Spannungsgewichtet und Penalty-Vektorprojektion nicht empfohlen, da sie sich an der vorherrschenden Spannungspotentiallinie orientieren und so, bei einem Weg durch das Feld, eine unnötige Zick-Zack-Line bilden würden. In **Abbildung 4.1.4** ist Penalty-Gradient-Linear im Vergleich zu den Optimierungsläufen von Gradient-Zurück in **Abbildung 4.1.3** zu sehen. Bei Penalty-Gradient-Linear nimmt der Anwender durch die Wahl der Schrittweitenverhältnisse keinen Einfluss auf die Richtung des Gradienten. Penalty normiert den Gradienten immer auf die Schrittweite eins. Trotzdem sind die Richtungen der beiden Algorithmen ähnlich. Wie schon im letzten Abschnitt erwähnt, ist die Änderung der Richtung bei Gradient-Zurück meist tendenziell.

Bei Penalty-Gradient-Linear sind in dem Diagramm auch jeweils die zwei Tastschritte zu sehen. Da er in der direkten Umgebung des Optimums auf die Sigmalinie trifft, ist keine Bewegung entlang derselben zu sehen, sondern nur eine kurze Kumulation.

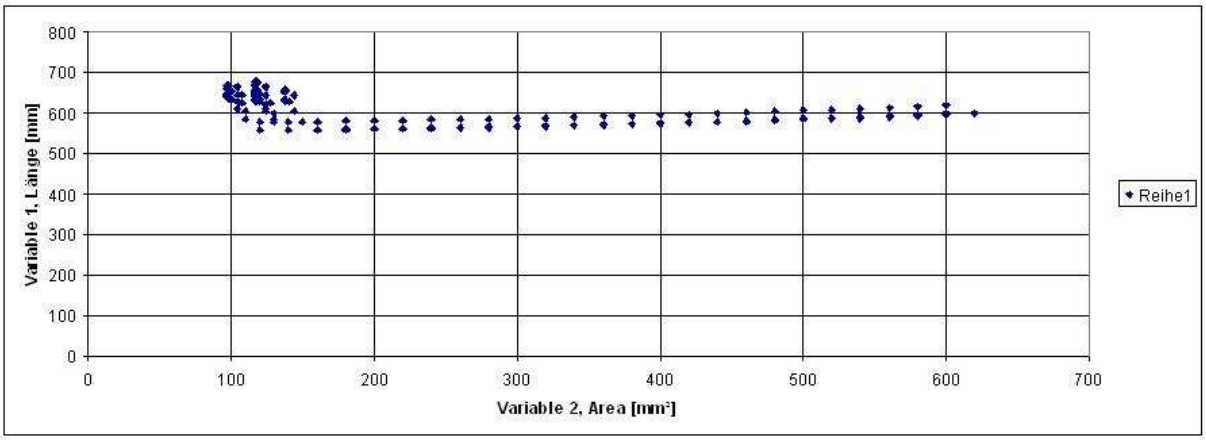


Abb. 4.1.4: Diagramm des Modells *2* Stäbe (Algorithmus Penalty)

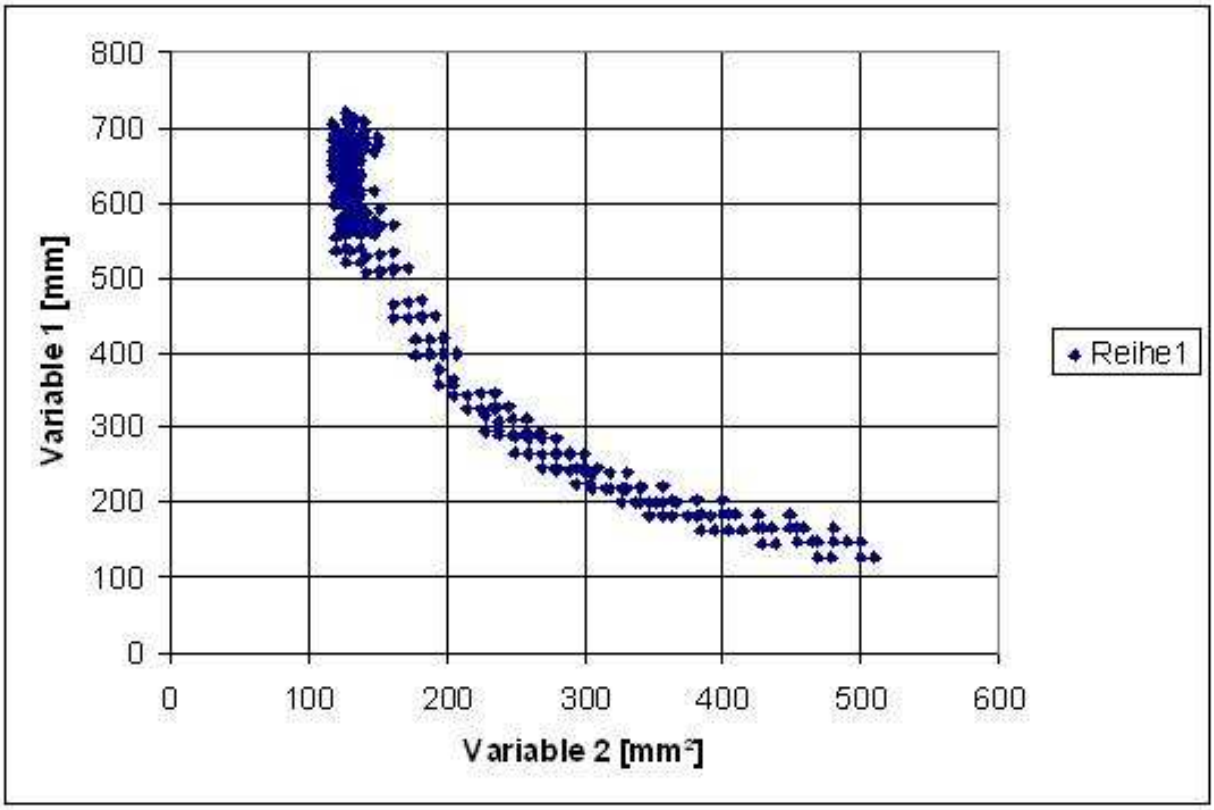


Abb. 4.1.5: Diagramm des Modells *2* Stäbe (Algorithmus Penalty-Spannungsgewichtet)

Anders stellt sich der Verlauf in der Optimierung mit Penalty-Spannungsgewichtet dar (**Abbildung 4.1.5**). Dieser startet an der Sigmalinie, aber in erheblicher Entfernung von dem Optimum. So kann er zeigen, wozu er ausgelegt ist. Seine Aufgabe ist es, an der Sigmalinie entlangzulaufen, da an dieser das Optimum zu erwarten ist. Außerdem zeigt er eine Fähigkeit, die Gradient-Zurück nicht hat. Er kann eine Variable so verändern, das diese Änderung allein das Ergebnis verschlechterte und gleichzeitig eine andere so verändern, dass in der Summe sich das Gesamtergebnis verbessert. Das bedeutet in diesem Fall, dass die Variable 1 vergrößert wird, und damit auch der Zielfunktionswert das Volumen, dass aber durch diese Vergrößerung die Spannung in dem Modell soweit sinkt, dass die Dicke der Stäbe soweit verringert werden kann, das dadurch das Volumen beider Variablenänderungen sinkt. Prinzipiell kann Penalty-Gradient-Linear das gleiche Ergebnis erzielen wie Penalty-Spannungsgewichtet. Der zweite Algorithmus ist aber durch die Gewichtung in der Lage, schneller an der Sigmalinie entlangzulaufen, da ihre jeweilige Steigung als Faktor in die Gradientenbildung einfließt. Dadurch wird der zur Linie orthogonale Anteil an der Bewegung minimiert. Trotzdem ist dieser Anteil noch vorhanden, da die Linie den Grund eines Tals darstellt, an dessen Ausrichtung sich der Algorithmus entlang arbeitet. Aber er verlässt immer wieder den unzulässigen Bereich und in der Nähe des Optimums lässt sich der beste zulässige Wert ablesen. Anders ist das bei Penalty-Vektorprojektion (**Abbildung 4.1.6**).

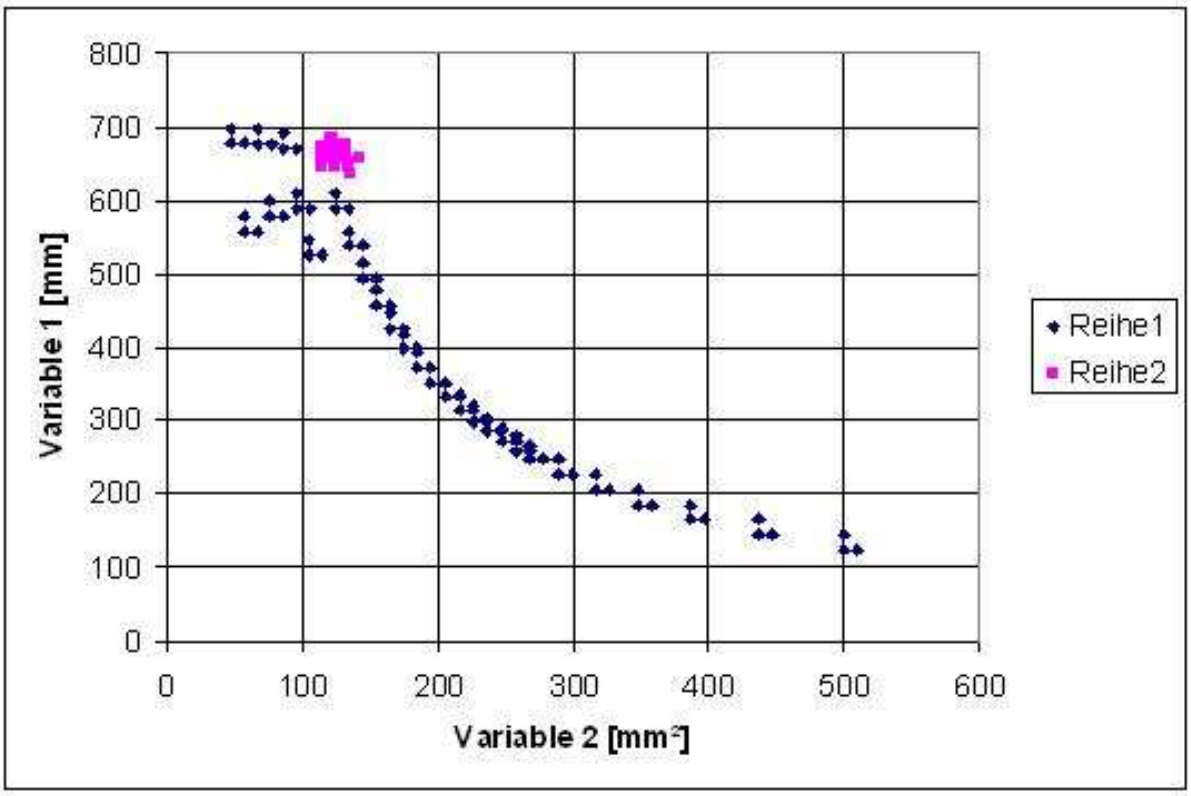


Abb. 4.1.6: Diagramm des Modells 2 Stäbe (Algorithmus Penalty-Vektorprojektion)

Dort findet keine Bewegung senkrecht zur Sigmalinie statt, da direkt auf sie projiziert wird. Dies hat zur Folge, dass die herrschende Konvexität des Feldes dazu führt, dass der Gradient früher oder später in den nichtzulässigen Bereich läuft und diesen dann nicht mehr verlässt. Die Richtung der Bewegung ist zwar noch richtig (parallel zur Linie) aber im unerlaubten Bereich. So verbessert sich zwar der Volumenwert, aber es werden keine erlaubten Verbesserungen festgestellt. Also muss man den Algorithmus rechnen lassen, bis keine erkennbare Parallelbewegung mehr festzustellen ist. Dann muss eine neue Rechnung mit Penalty-Gradient-Linear gestartet werden (Reihe 2). Dieser ist in der Lage in einem gewissen Rahmen, aus der Tiefe des unzulässigen Bereichs in den zulässigen Bereich zu laufen. Dies wird dann in der Nähe des Optimums sein.

4.1.4 Vektorprojektion-Sigmakorrektur

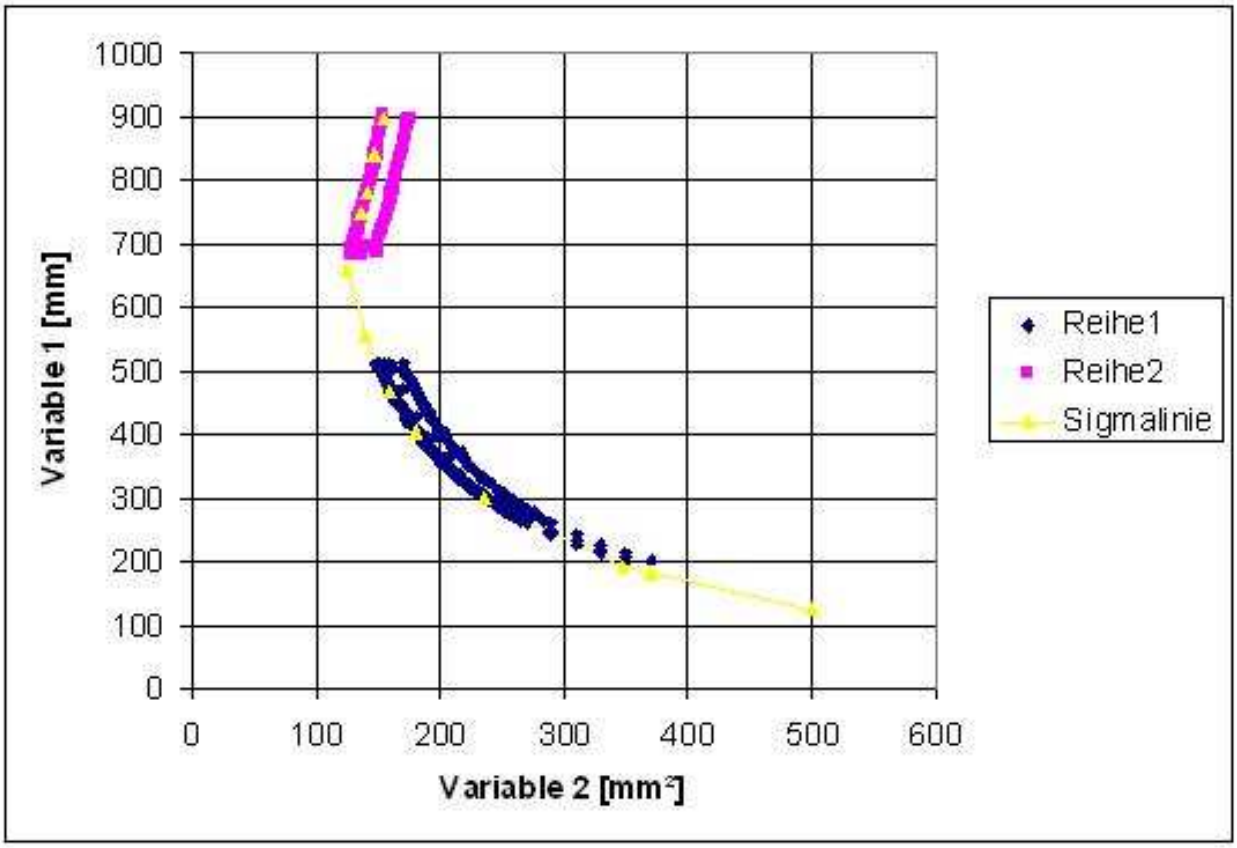


Abb. 4.1.7: Diagramm des Modells 2 Stäbe (Algorithmus Vektorprojektion-Sigma-korrektur)

Im Diagramm von **Abbildung 4.1.7** sind neben der Sigmalinie zwei Rechenläufe des Algorithmus Vektorprojektion-Sigmakorrektur aufgeführt. Diese beiden Rechenläufe haben unterschiedliche Startwerte, welche aber jeweils nahe an der Sigmalinie liegen. Dieser Algorithmus ist darauf spezialisiert, die Sigmalinie abzugehen. Im Unterschied zu Penalty-Vektorprojektion stellt er fest, wenn der zulässige Bereich verlassen wurde, und nutzt den Spannungsgradienten dazu, diesen Bereich wieder zu erreichen. Da kein besserer Wert als der Endwert der Reihe2 erreicht wurde, ist anzunehmen, dass es sich dabei um das globale Minimum handelt. Somit hat Reihe2 keine Veranlassung seinen Weg fortzusetzen. Reihe1 hingegen ist suboptimal und das erreichte Ergebnis ist ein größerer Wert (Reihe2: 283 cm^3, Reihe 1: 318,4 cm^3). Das liegt daran, dass die Genauigkeit der Sigmakorrektur zu grob eingestellt war. Diese hätte einfach noch weitere Korrekturschrittverkleinerungen benötigt, um von der Seite des zulässigen Bereichs näher an die Sigmalinie zu gelangen, dort einen verbesserten Wert zu ermitteln und von diesem aus die Optimierung fortzusetzen.

4.1.5 Der Algorithmus Plus-Minus

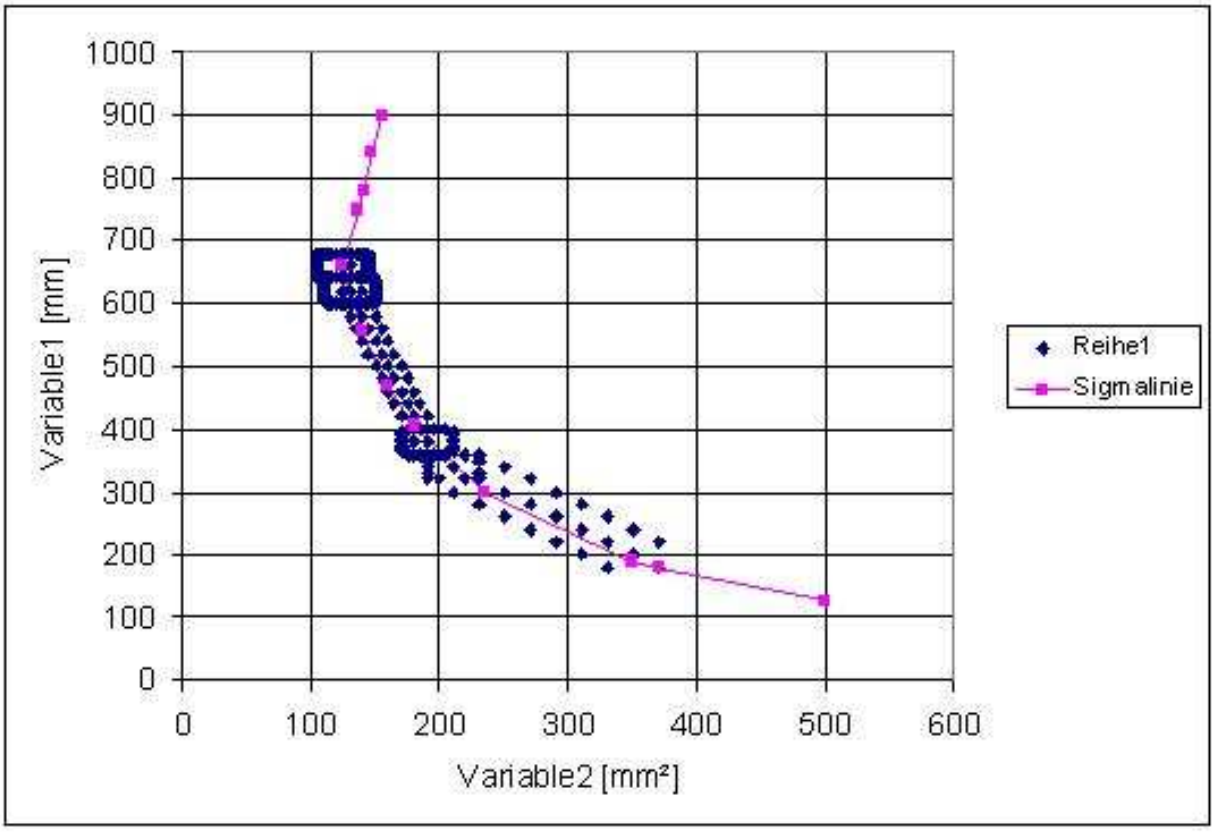


Abb. 4.1.8: Diagramm des Modells 2 Stäbe (Algorithmus Plus-Minus)

Auch der Algorithmus Plus-Minus ist in der Lage, das Minimum zu lokalisieren (**Abbildung 4.1.8**). Hier ist gut zu sehen, dass mehr Rechnungen nötig sind, wenn ein besserer Wert nicht sofort gefunden wird. Dies ist der Fall, wenn die Rechtecke, die gebildet werden durch mehr Rechnungen ausgefüllt werden und eindeutiger als Rechteck zu erkennen sind. Dies ist um den Wert 190mm²/390mm und 130mm²/620mm der Fall. Das Rechteck um 120mm²/670mm hat keinen Erfolg mehr gebracht. Bei allen anderen Rechtecken wurde ein verbesserter Wert so schnell gefunden, dass sie teilweise nicht als Rechtecke zu erkennen sind. Auch bei diesem Rechenlauf war die Schrittweite so groß gewählt, dass nicht an dem lokalen Minimum die Rechnung beendet wurde.

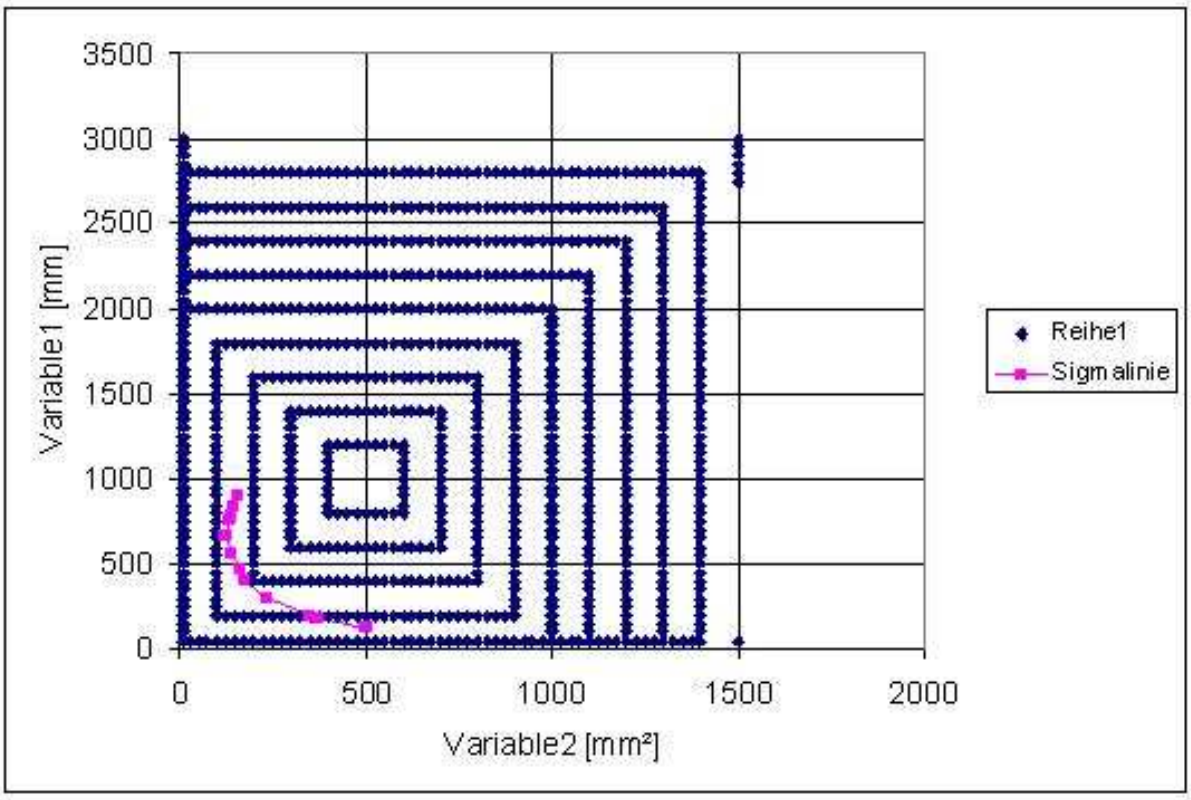


Abb. 4.1.9: Diagramm des Modells 2 Stäbe (Algorithmus Plus-Minus-Global)

Der Algorithmus Plus-Minus-Global hat den Anspruch, auf der Suche nach dem globalen Optimum unterstützen zu können. Dies tut er auf zwei Weisen. In **Abbildung 4.1.9** ist eine Möglichkeit durchgeführt worden. Hier wird der Bereich, der interessant ist, sehr ausgedehnt und untersucht. So wird mit sehr großen Schrittweiten ein großer Bereich berechnet, um festzustellen, ob nicht an anderen als den untersuchten Stellen im Feld, ein kleineres Minimum , als das bisher gefundene, auftritt, oder einen Bereich aufzeigt, der genauer untersucht werden sollte. Solche Werte wurden in diesem Beispiel allerdings nicht gefunden. Alternativ kann dieser Algorithmus auch als erste Rechnung gewählt werden, um so den zu erwartenden Bereich des Zielfunktionswertes abzuschätzen und die Startwerte festzulegen.

Die zweite Fähigkeit, die diesem Algorithmus die Globalität zuschreibt, ist die Schrittweitenvergrößerung. Zur Überprüfung kann an einem gefundenen Optimum dieser Algorithmus neu gestartet werden. Sollte es ein globales Minimum sein, könnte Plus-Minus-Global dieses wieder verlassen und hinter dem *Berg* neue Minima finden. Der Nachteil dieser Vorgehensweise ist klar zu sehen, ist aber bei jeder globalen Untersuchung nicht zu vermeiden. Dies ist der große Zeitaufwand. Im Gegensatz zu lokalen Suchen, die nach klaren Regeln schnell zum Ziel führen können, ist die Richtung der Globalen Suche kaum vorherzusagen und meist eher zufällig.

4.2 Das Modell „Kraftteilung2Var“

4.2.1 Der Modellaufbau

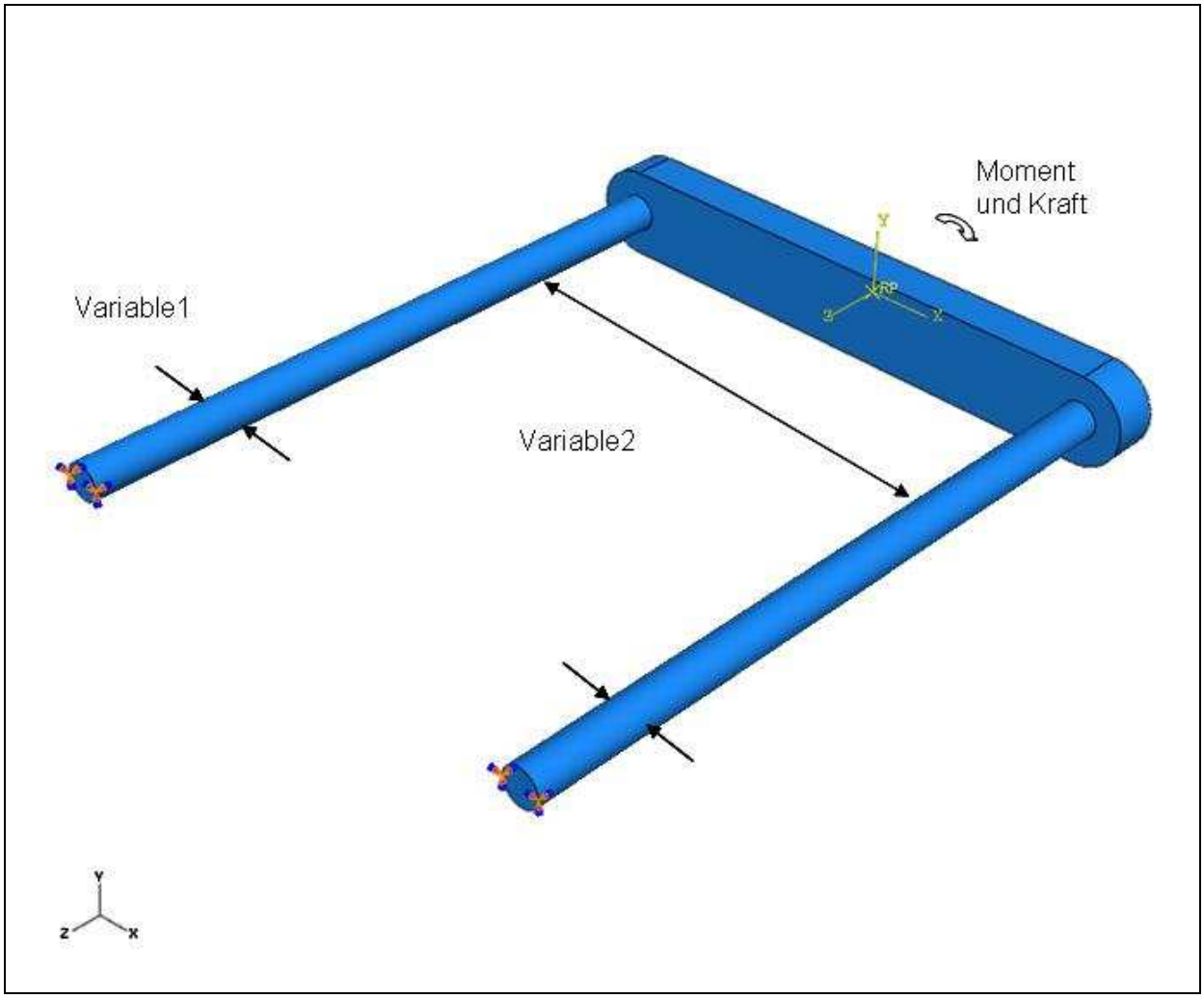


Abb. 4.2.1: Das Modell Kraftteilung2Var

Das Modell besteht aus einer Platte und zwei Stäben mit denen Kräfte und Momente übertragen werden. Bei der Erstellung des Modells wurden *Solids* verwendet (**Abbildung 4.2.1**). Im Sketcher wurde die Mittelplatte entworfen und danach wurde ein *Shape/Extrude* durchgeführt (**Abbildung 4.2.2**). Im ersten Sketcher wurde die Variable 2 erstellt durch Bemaßung der Länge (120 mm). Dabei ist der Achsabstand der beiden Stäbe maßgebend. Im Sketcher 2 wurden die beiden Stäbe platziert, extrudiert und bemaßt (8.4 mm). Die darauf folgende Vernetzung nutzte hexagonale Elemente.

Die unteren Enden der Stäbe sind fest eingespannt und der Referenzpunkt ist mit der oberen Fläche der Mittelplatte verbunden. Dieser Referenzpunkt wird dann mit Kräften und Momenten beaufschlagt. Im ersten Fall wird ein Moment um die Y-Achse von 20 kNm angegeben.

In diesem Fall ist das entstandene Optimierungsproblem anschaulich. Wird die Variable2 verkleinert, sinkt zwar der Volumenwert, aber die Belastung der Stäbe steigt, so dass sie verstärkt werden müssen und das Volumen größer wird.

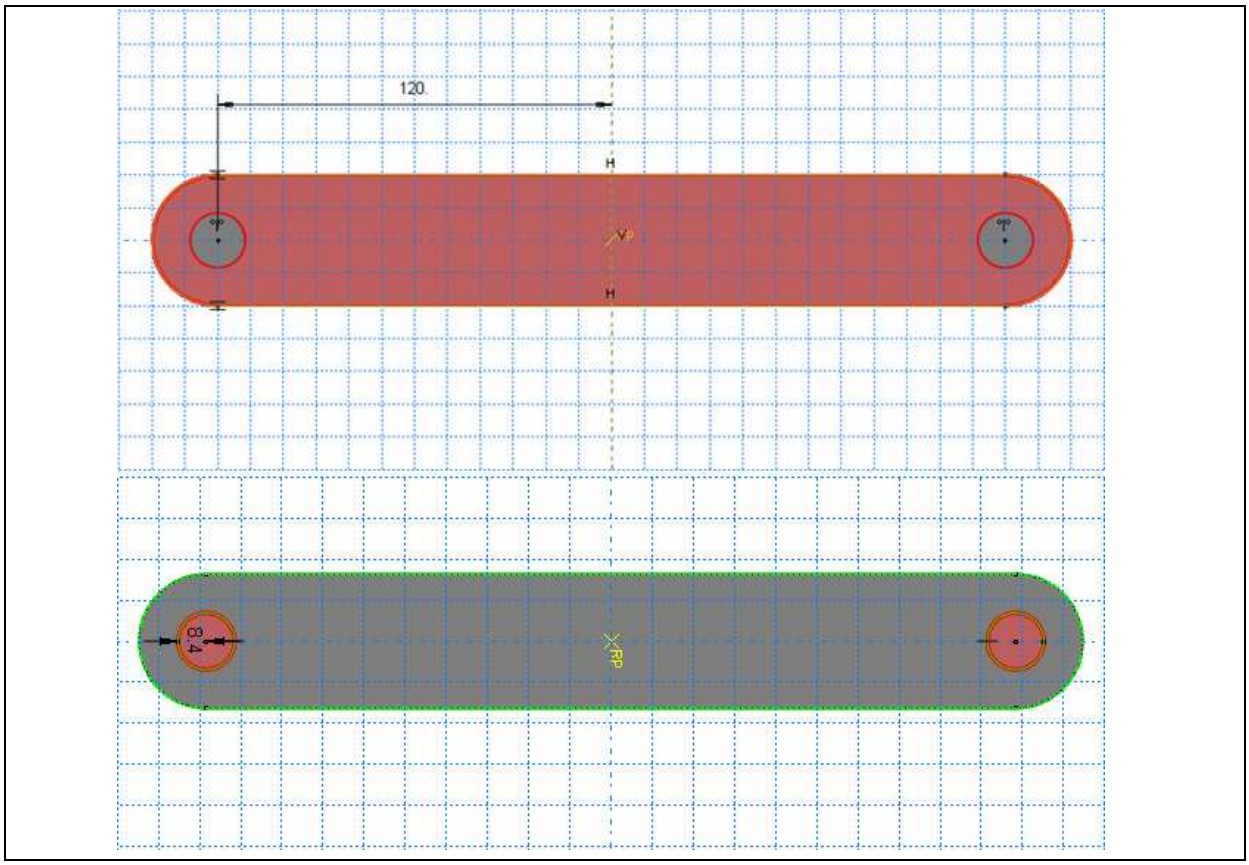


Abb. 4.2.2: Das Modell Kraftteilung2Var im Sketcher1 und Sketcher2

4.2.2 Die Rechenläufe

In **Abbildung 4.2.3** sind die Rechnungsverläufe bei unterschiedlichen Startwerten und unterschiedlichen Schrittweitenverhältnissen (Startwert1/b) zu sehen. Auch sind zwei Äquipotentiallinien des Volumenfeldes eingezeichnet. Das typische Abknicken des Verlaufes bei dem ersten Kontakt mit der Sigmalinie ist sehr deutlich zu sehen. Weiterhin ist hier erneut der Nachteil des Algorithmus Gradient-Zirück zu erkennen. Die Rechnung von Startwert 3 (Stw3) erreicht die Sigmalinie und kann von diesem Punkt aus keine Verbesserungen mehr erzielen, da die Variable 1 dazu vergrößert werden müsste. Deshalb sollte von diesem Punkt aus eine weitere Rechnung mit anderen Algorithmen gestartet werden, was im nächsten Abschnitt beschrieben wird.

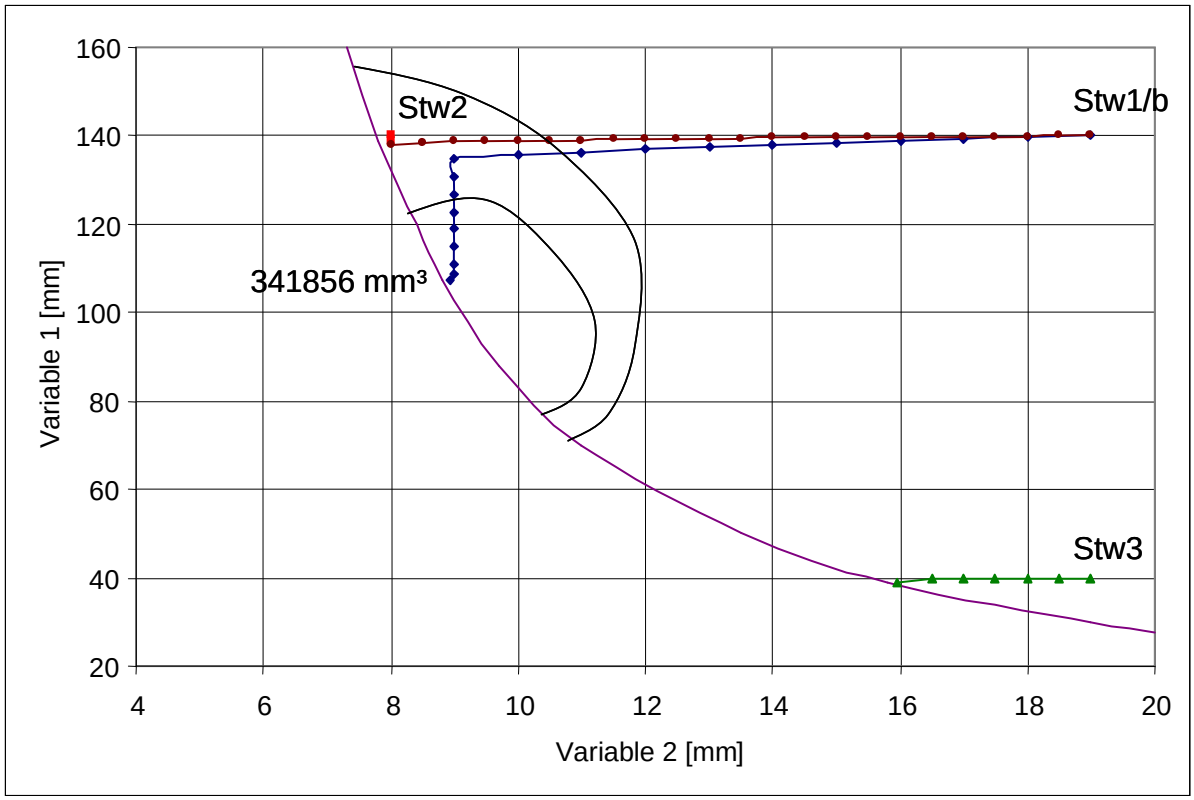


Abb. 4.2.3: Diagramm des Modells Kraftteilung2Var (Algorithmus Gradient-Zurück)

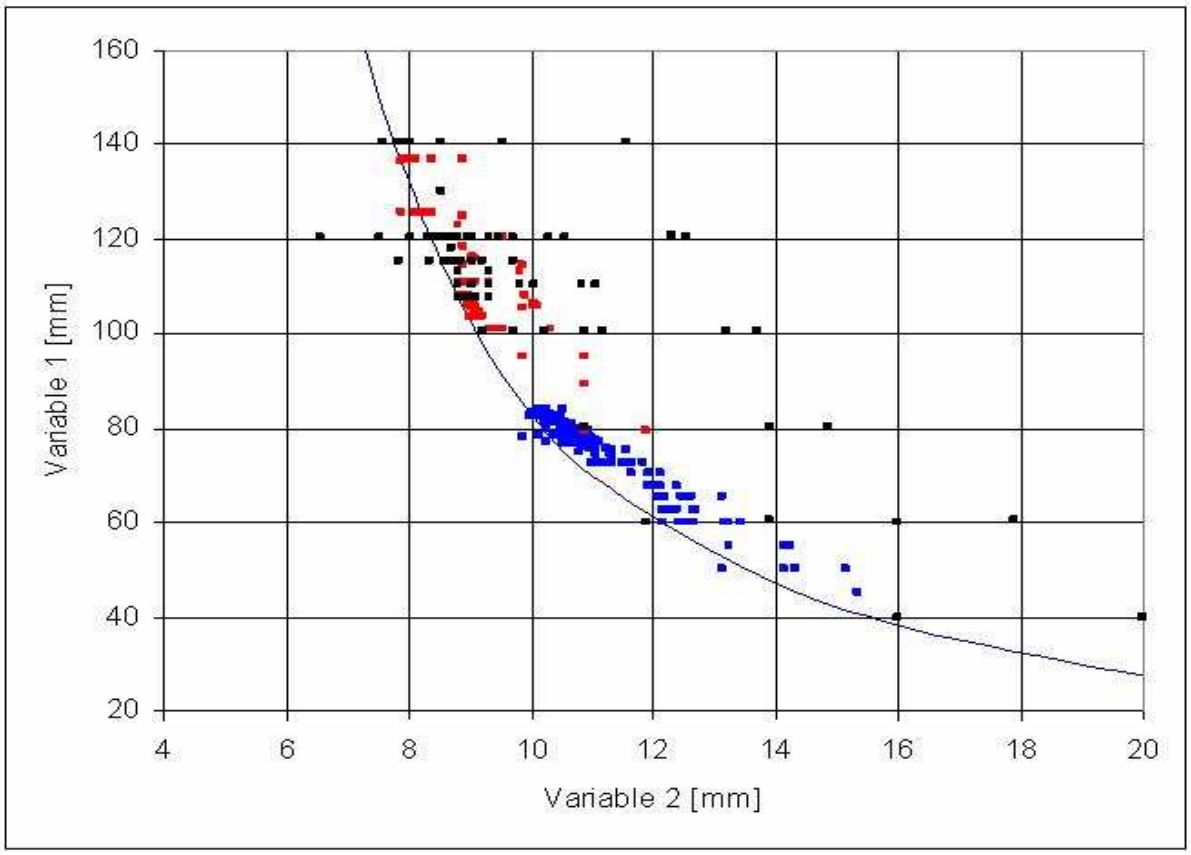


Abb. 4.2.4: Kraftteilung2Var (Vektorprojektion-Sigmakorrektur)

An den Endwerten der Reihe Stw3 wird eine neue Rechnung mit Vektorprojektion-Sigmakorrektur gestartet (**Abbildung 4.2.4**, Blaue Punkte). Diese Rechnung erreicht das Minimum jedoch nicht. Eine neue Rechnung (Rote Punkte) in deren Endwert mit größerer Schrittweite lässt ein neues Optimum erreichen. Eine dritte Rechnung (Schwarze Punkte), die auch im Endwert der Rechnung Stw3 gestartet wird, erhält sehr große Schrittweiten. In diesem Feld erhält man damit das gleiche optimale Ergebnis wie die zweite Rechnung (Rote Punkte). Proberechnungen in durch die Optimierungsläufe nicht erfassten Gebieten ergaben, dass das ermittelte Optimum das globale Optimum ist. Die Begründung dafür, dass die blaue Linie das Minimum nicht erreicht, liegt an der Neuvernetzung des Bauteils nach jeder Änderung der Geometrie. Diese Änderung hat, da immer mit der gleichen Elementkantenlänge gerechnet wird, einen sehr geringen Einfluss auf die errechnete Spannung. Im Fall der kubisch vernetzten Stäbe dieses Beispiels kann es jedoch geschehen, dass die Neuvernetzung doch einen Einfluss ausübt. So wird bei einem bestimmten Durchmesser der Stäbe der Einfluss der Variable 2 auf die Spannung geringfügig geändert. Dies hat jedoch die Wirkung, dass die ermittelten Gradienten und Ausführungsschritte nicht mehr erfolgreich genug sind. Es erfolgen zu früh zu viele Schrittweitenhalbierungen und die Optimierung läuft sich kurz vor dem Optimum fest. Dieser Einfluss der Vernetzung kann jedoch umgangen werden. Wie in Abbildung 4.2.4 dargestellt ist, sind andere Schrittweiten eine Möglichkeit (schwarze Punkte). Diese Schrittweiten müssen nicht so sehr von der Rechnung der blauen Punkte abweichen, wie in der Abbildung. Schon eine sehr geringe Änderung des Schrittweitenverhältnisses der beiden Variablen, hat den negativen Effekt der Neuvernetzung verhindert.

In **Abbildung 4.2.5** sind die Spannungsverläufe des Modells im Optimum bei 8.9mm/108mm zu sehen.

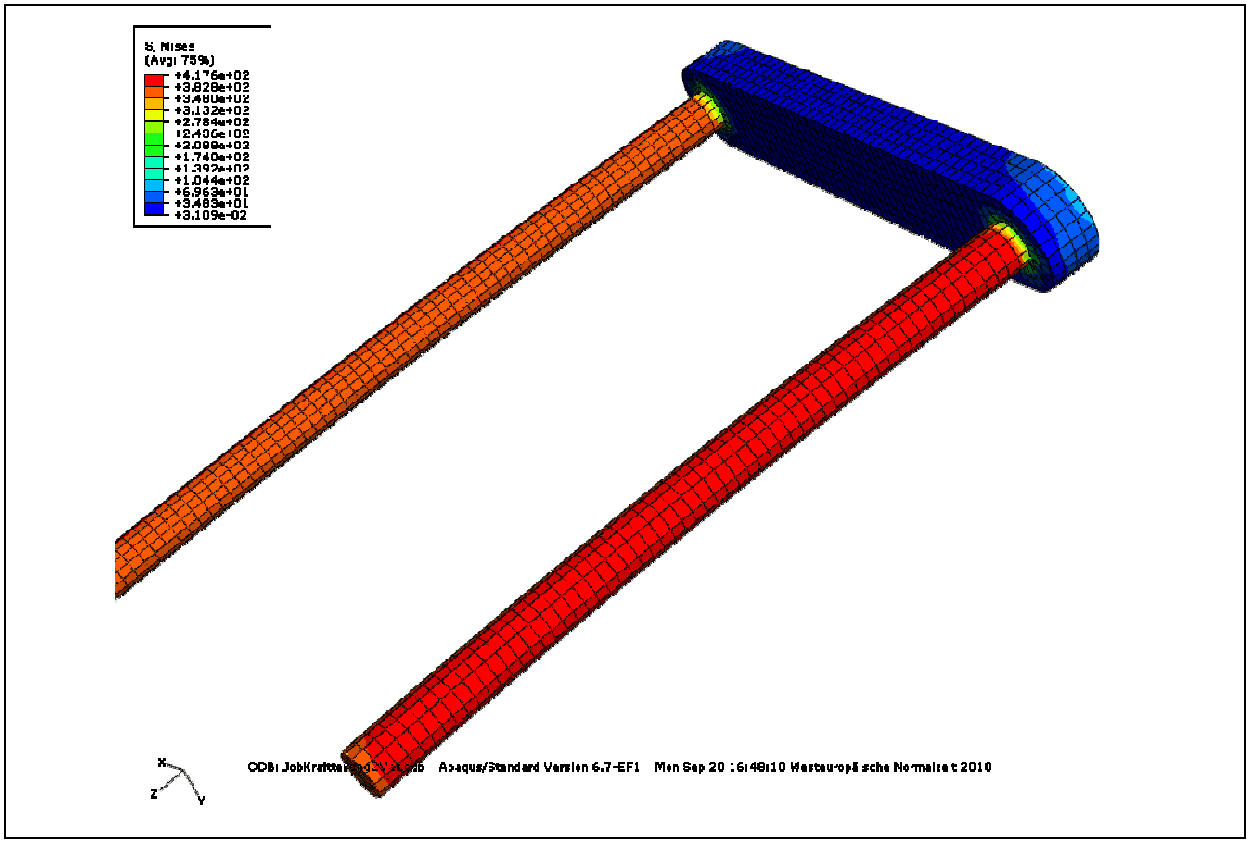


Abb. 4.2.5: Ergebnis Kraftteilung2Var

4.3 Das Modell „Momentenplatte2Var“

4.3.1 Der Modellaufbau

Das Modell besteht aus einer schmalen Platte, mit der Kräfte übertragen werden. Da sie an ihrer einen Seite fest eingespannt ist, erzeugt die Kraft ein großes Moment, woher der Name des Modells kommt. Die Platte ist an dem einen Ende fest eingespannt und an dem anderen Ende mit einer Kraft belastet. Diese besteht aus den Anteilen F_x = 10 kN und F_y = 10 kN. Der x-Anteil der Platte beträgt immer 500 mm. In der Ausgangsposition ist das die Länge der Platte. Sie ist 60 mm breit. Ihr freies Ende ist verschieblich, das bedeutet, sie wird mit der Änderung der Variable1 um ihr eingespanntes Ende gekippt. Ihre Länge ändert sich dabei entsprechend der Winkelbeziehungen. In der Anfangsposition ist die Variable1 = 500 mm. Die Variable2 ist die Dicke der Platte (**Abbildung 4.3.1**). Das Modell ist ein dreidimensionales Schalenmodel. Das bedeutet, es ist von einer geringen Dicke gegenüber den anderen Maßen auszugehen. Die Geometrie wird zweidimensional gestaltet und mit einer Elementdicke beaufschlagt.

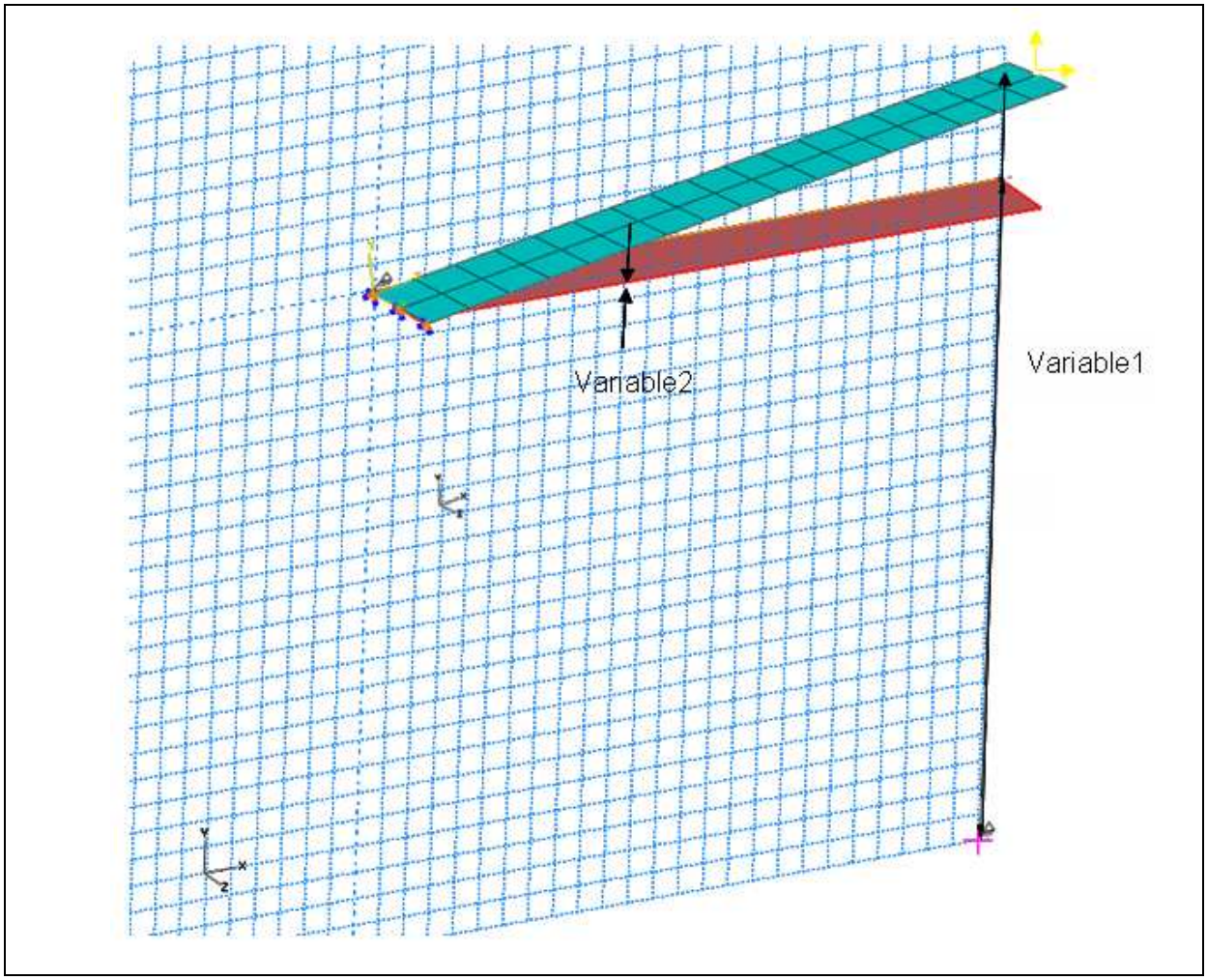


Abb. 4.3.1: Das Modell Momentenplatte2Var

Zum Ablauf der Optimierung lässt sich sagen, dass bei einer Vergrößerung der Variablen1 das Volumen des Körpers vergrößert wird. Gleichzeitig wird der zur Platte orthogonale Anteil der Kraft geringer (bei der in diesem Beispiel gewählten Kraft) und dadurch auch das durch die Kraft erzeugte Biegemoment, als dessen Folge die Dicke der Platte verringert werden kann. Aus diesem Zusammenhang entsteht das Optimum, das zu finden ist.

4.3.2 Die Rechenläufe

Es werden die Rechenläufe des Algorithmus Vektorprojektion-Sigmakorrektur durchgeführt (**Abbildung 4.3.2**). Folgend sind die Ergebnisse zweier Rechnungen zu sehen. Die Rechnung R1 wurde bei den Werten 35mm/500mm und R2 wurde bei 35mm/1500mm gestartet. Es ist ein eindeutiges Minimum bei Variable 1 = 1000 zu sehen. Beide Startwerte boten keine Schwierigkeit das Minimum zu finden. Sehr sauber läuft der Algorithmus entlang der Sigmaline.

Eine weitere Rechnung muss erwähnt werden. Der Algorithmus Penalty-Spannungsgewichtet hat keine Schwierigkeiten mit dem zu lösenden Problem. Er bringt das gleiche Ergebnis wie R2. Allerdings ist er spürbar langsamer. Dies geschieht erst in Annäherung an das Minimum. Bis dahin ist er ähnlich schnell.

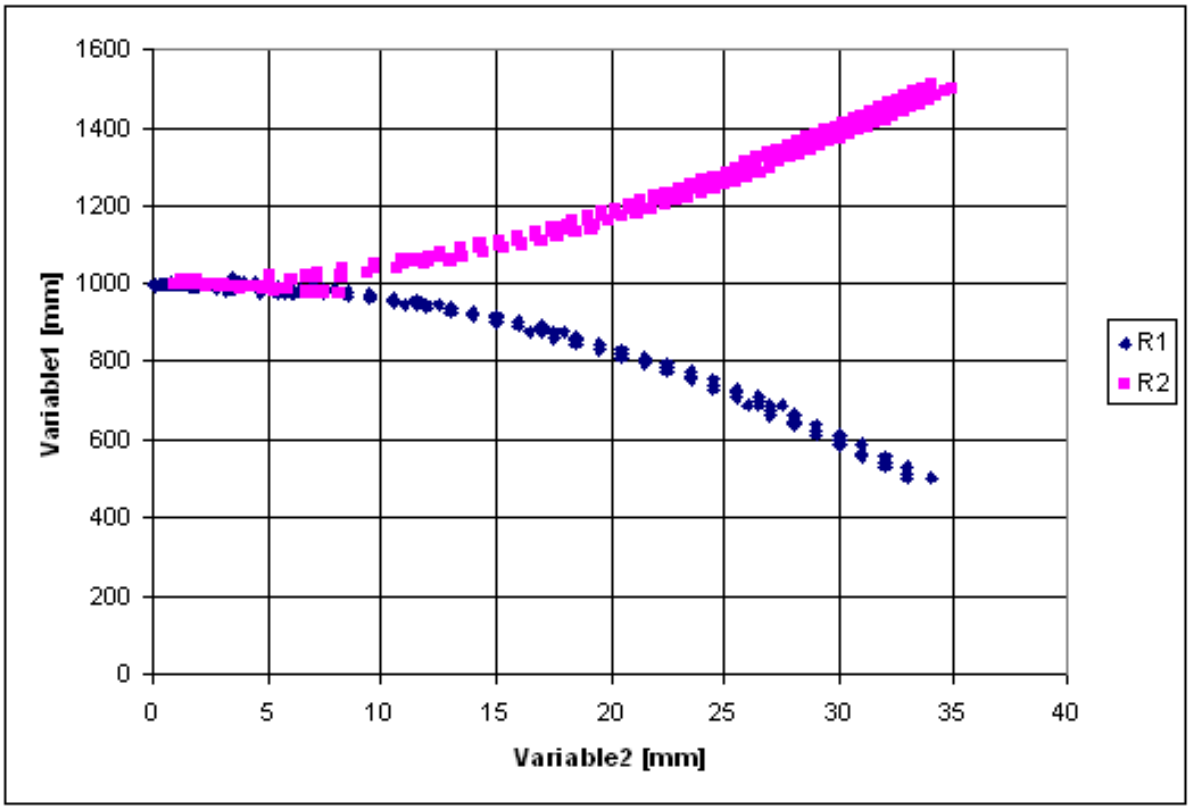


Abb. 4.3.2: Diagramm des Modells Momentenplatte2Var (Algorithmus Vektor-projektion-Sigmakorrektur)

4.4 Das Modell „Würfel“

4.4.1 Der Modellaufbau

Der Würfel wird zur Kraft- und Momentenübertragung genutzt. Er ist an seiner unteren Seite fest eingespannt und die obere Seite wird mit Kräften und Momenten belastet. In seiner Ausgangsposition ist er ein Würfel. Daher der Name des Modells. Je nach der Belastung, wird sich ein Quader mit unterschiedlichen Seitenlängen herausbilden. In **Abbildung 4.4.2** sind die Variable 1 = 100 mm und die Variable 2 = 300 mm. Die beiden Variablen des Modells sind die Seitenlängen des Würfels. Die Belastungen sind:

F_x	4000 kN	M_x	1000 kNm
F_y	2000 kN	M_y	1000 kNm
F_z	1000 kN	M_z	0

Abb. 4.4.1: Belastungen

Das Optimierungsproblem ist die Gewichtsminimierung mit Spannungsrestriktion. Es wird also die Möglichkeit gesucht, die so nah wie möglich am Würfel ist, aber auch die Spannungsbedingung einhält. Bei den gegebenen Belastungen, lässt sich vermuten, das Variable 2 am Ende größer sein wird, als Variable eins. Dies wird durch die Rechnungen bestätigt.

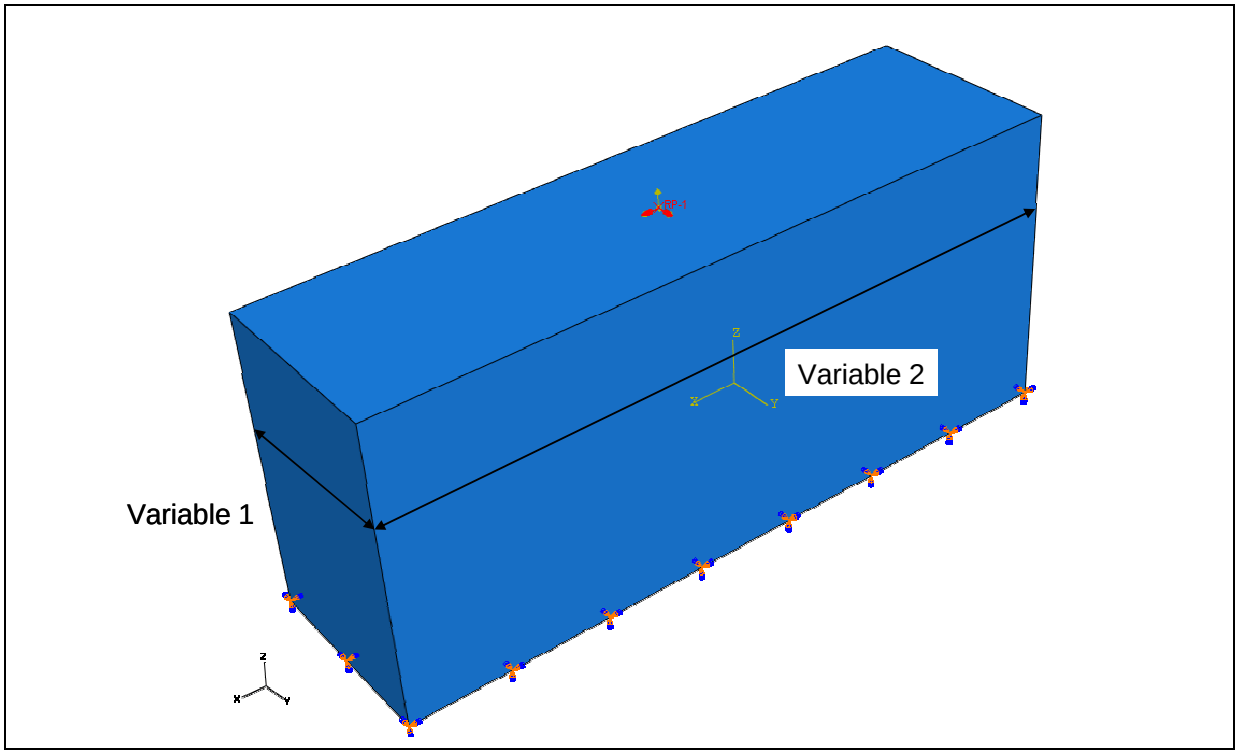


Abb. 4.4.2: Das Modell Wuerfel

4.4.2 Die Rechenläufe

Die Startwerte für GraZu1 (Gradient-Zurück) von 500mm/800mm und Pe (Penalty-Gradient-Linear) von 600mm/600mm sind frei gewählt. Eine Bedingung ist allerdings, dass sie im zulässigen Bereich liegen müssen. GraZu1 gelangt mit einer gewissen Anzahl an Rechnungen an die Sigmalinie und erreicht dort nach weiteren Rechnungen sein Minimum. Ein besseres Ergebnis ist nicht möglich, da er Variable 2 nicht verschlechtern kann. Anders ist es bei Pe. Dieser hat die Möglichkeit an der Sigmalinie weiterzugehen. Dies ist jedoch mit recht vielen Rechnungen verbunden.

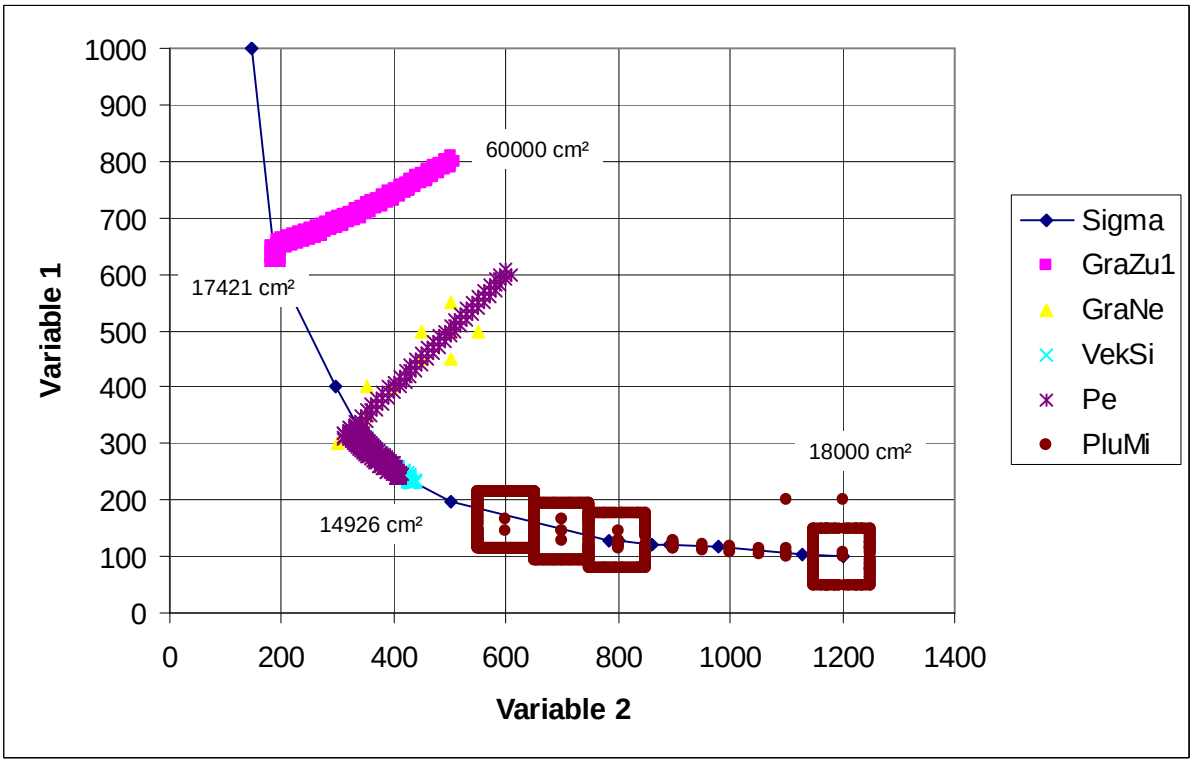


Abb. 4.4.3: Rechenläufe

Hinter Pe ist die Rechnung GraNe (Gradient-Nebenbedingung) leicht verborgen. Trotzdem ist gut zu sehen, dass der Algorithmus mit sehr wenigen Rechnungen die Sigmalinie erreicht, dort allerdings keine Verbesserung mehr erreichen kann. Folgend wird an dieser Stelle die Rechnung VekSi (Vektorprojektion- Sigmakorrektur) gestartet. Dies wird von Pe verdeckt und ist erst bei Nichtüberschneidung zu sehen. Bei dem nicht im Bild zu sehenden Startwert 100mm/2000mm wurde eine weitere Vektorprojektion-Sigmakorrektur-Rechnung gestartet. Dies ist im Bild nicht dargestellt, aber dieser Lauf geht entlang der Sigmalinie bis zum Optimum bei 500mm/200mm, welches etwas besser ist als der Wert von VekSi. Den besten Wert erreicht allerdings PluMi (Plus-Minus) bei 550mm/180mm mit 14888 cm². Die Wertepaare richten sich nach dem Diagramm und nicht nach der Variablennummer, das heißt, zuerst wird der Abszissenwert genannt.

238.9421285	418.9177198	'Ergebniswert: '	15014563.54	'Ergebnis: '	437.8807373	'Index: '	3	'Gesamtindex: '	34	'Nummer: '	34	'	'Vneuneu1'	5.031016541	'Vneuneu2'	-10
248.9421285	418.9177198	'Ergebniswert: '	15642940.15	'Ergebnis: '	413.3952332	'Index: '	3	'Gesamtindex: '	34	'Nummer: '	35					
238.9421285	428.9177198	'Ergebniswert: '	15372976.8	'Ergebnis: '	422.1748962	'Index: '	3	'Gesamtindex: '	34	'Nummer: '	36					
232.527786	428.9177198	'Ergebniswert: '	14960293.25	'Ergebnis: '	438.6700439	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	37	'	'Vneuneu1'	-6.41434253	'Vneuneu2'	10
242.527786	428.9177198	'Ergebniswert: '	15603669.93	'Ergebnis: '	413.4162903	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	38					
232.527786	438.9177198	'Ergebniswert: '	15309085.08	'Ergebnis: '	426.0438232	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	39					
237.5275262	418.9177198	'Ergebniswert: '	14925673.64	'Ergebnis: '	441.5481567	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	40	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
247.5275262	423.91746	'Ergebniswert: '	15739685.66	'Ergebnis: '	409.1002808	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	41	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
242.5275262	421.4175899	'Ergebniswert: '	15330805.46	'Ergebnis: '	424.8783264	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	42	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
237.5275262	418.9177198	'Ergebniswert: '	14925673.64	'Ergebnis: '	441.5481567	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	43	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
240.0275262	420.1676548	'Ergebniswert: '	15127770.78	'Ergebnis: '	433.0976563	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	44	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
238.7775262	419.5426873	'Ergebniswert: '	15026605.02	'Ergebnis: '	437.293457	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	45	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
237.5275262	418.9177198	'Ergebniswert: '	14925673.64	'Ergebnis: '	441.5481567	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	46	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
238.1525262	419.2302035	'Ergebniswert: '	14976109.53	'Ergebnis: '	439.4134216	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	47	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
237.8400262	419.0739617	'Ergebniswert: '	14950884.81	'Ergebnis: '	440.4789124	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	48	'	'Vneuneu1'	4.999740186	'Vneuneu2'	-10
237.527786	428.9177198	'Ergebniswert: '	15281981.77	'Ergebnis: '	425.7186279	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	49					
232.527786	433.9177198	'Ergebniswert: '	15134689.18	'Ergebnis: '	433.6273804	'Index: '	3	'Gesamtindex: '	37	'Nummer: '	50					

Abb. 4.4.4: Auszug aus der Ergebnisliste

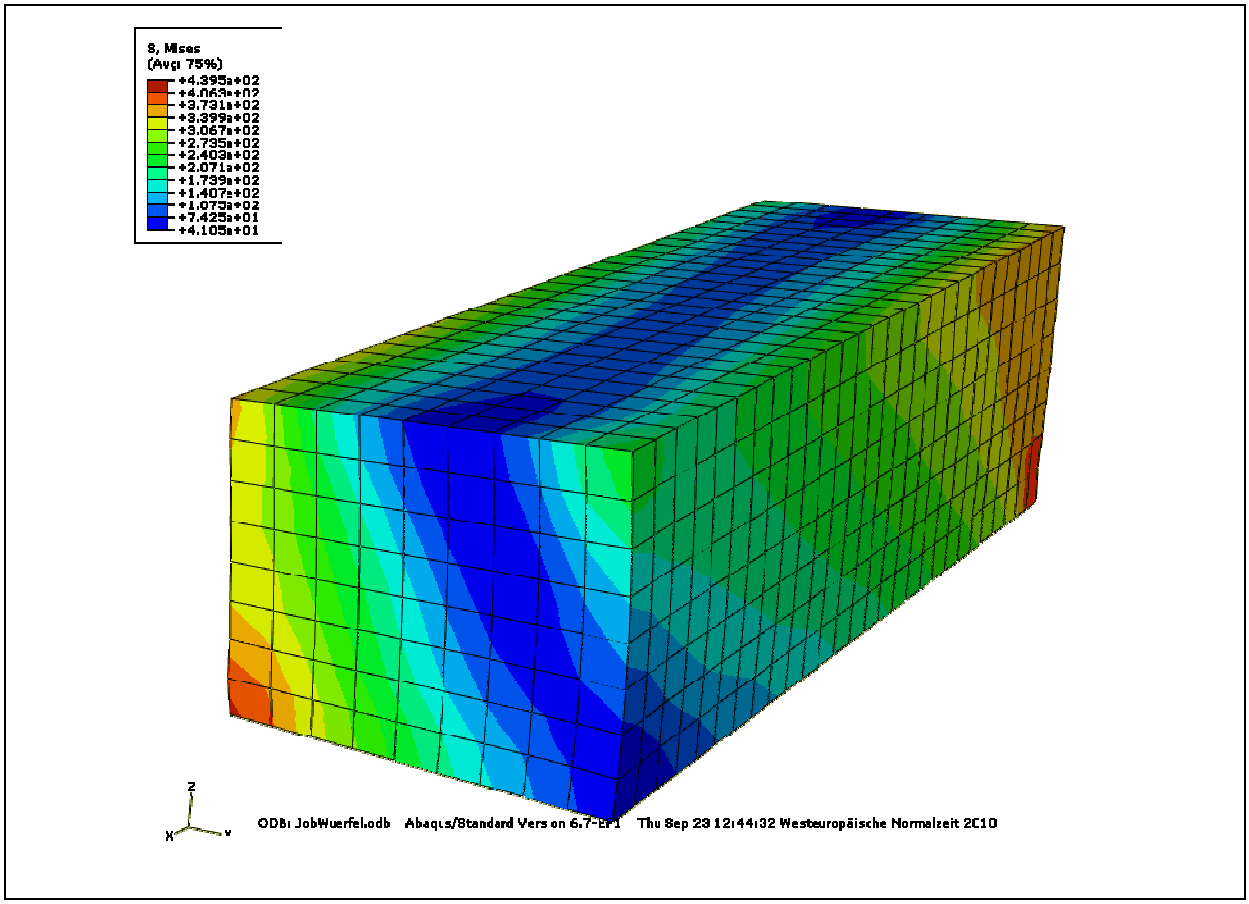


Abb. 4.4.5: Spannungsplot des Optimums

Abbildung 4.4.4 ist ein Auszug aus der Ergebnisliste der Rechnung VekSi der Rechnungsnummer 34 bis 50. Zeile 34 ist ein erfolgreich durchgeführter Ausführungsschritt. Es folgen zwei Tastschritte mit den Schrittweiten 10 mm. Der daraus ermittelte Vektor ist -6,41 mm und 10 mm. Dieser Vektor ist der Ausführungsschritt und wird zu Zeile 34 addiert. Er ist erfolgreich, was durch die Änderung des Gesamtindex zu sehen ist. Der danach ermittelte Ausführungsschritt verletzt die Restriktion. Daraufhin greift der Spannungsvektor, der Sp_x =10 mm und Sp_y = 5 mm beträgt. Dieser Wert wird nun bis zu achtmal (im Optimierungsfenster eingegebener Wert) um 180° gedreht und halbiert. In diesem Fall bringt dies jedoch keinen Erfolg (verbesserter zulässiger Wert). So springt der Algorithmus auf 37 zurück (bester Wert) und startet mit halbierter Schrittweite einen neuen Tastzyklus.

Abbildung 4.4.5 ist der Spannungsplot des Modells bei dem ermittelten Optimum bei den Längen Variable 1 = 180,46 mm und Variable 2 = 550 mm.

4.5 Das Modell „3 Stäbe“

4.5.1 Der Modellaufbau

Das Modell besteht aus drei Stäben, die der Kraftübertragung dienen. Die linken Enden der drei Stäbe sind unverschieblich und die rechten, miteinander verbundenen Enden sind durch die Kräfte F_x =10 kN und F_y = 2 kN belastet. Die Variablen eins bis drei sind die Querschnittsflächen der einzelnen Stäbe. Die Länge des unteren Stabes ist 1000 mm. Das Optimierungsproblem ist die Gewichtsoptimierung mit Spannungsrestriktion. Ziel ist also die Konstellation der Dicken der Stäbe zu erhalten, die möglichst leicht ist und dabei die Spannung nicht verletzt (**Abbildung 4.5.1**). Das Modell besteht aus Truss-Elememten und ist linear-statisch.

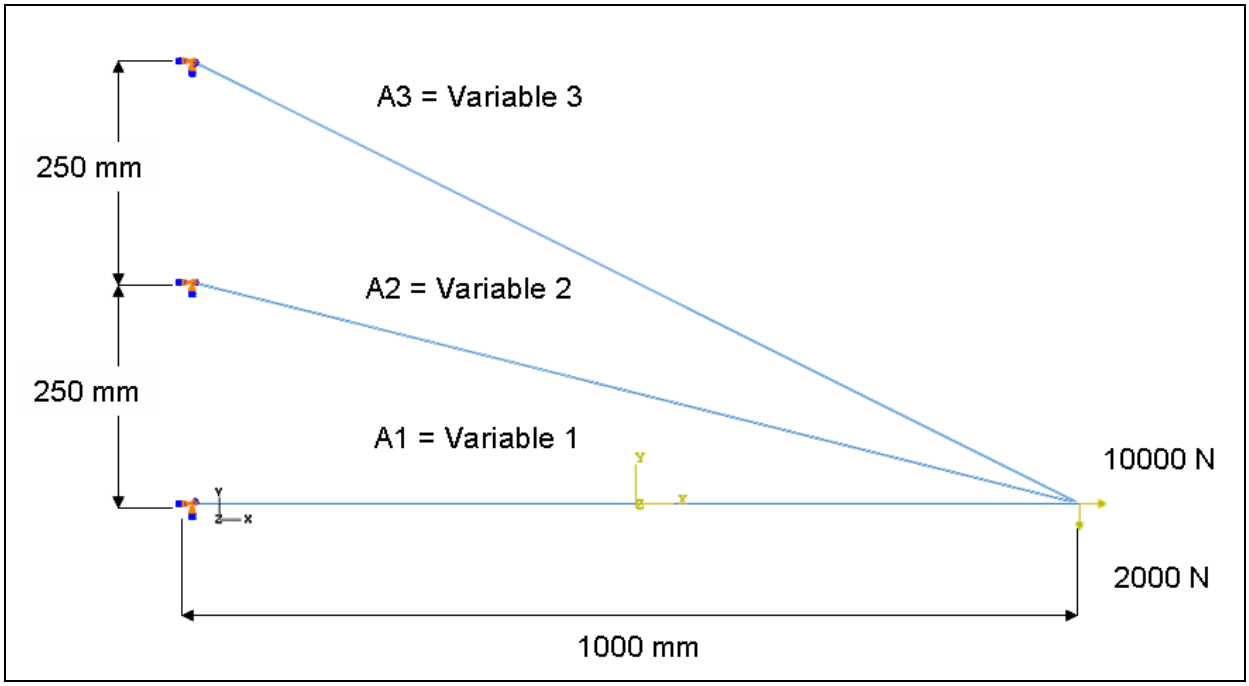


Abb. 4.5.1: Das Modell 3 Stäbe

4.5.2 Die Rechenläufe

Die Startwerte der ersten drei Rechnungen (**Abbildung 4.5.2**), die Rechnungen Gradient und Gradient-Zurück 1 und 2, waren frei gewählt. Sowohl die Konstellationen der Ergebniswerte der Variablen, als auch die jeweiligen Ergebniswerte und Ergebnisse und auch die Anzahl der benötigten Rechnungen, die Nummern, sind in ähnlichen Bereichen. Aber auch der in der Nähe der Ergebnisse der ersten Läufe gestartete Lauf Vektor-Sigmakorrektur, kann diesen Bereich nicht verlassen. Dies spricht dafür, dass sich dort ein lokales Minimum befindet.

		V1 [mm²]	V2 [mm²]	V3 [mm²]	Ergebnis-wert [mm³]	Ergebnis [N/mm²]	Einzelrechnungen
GradientStartwert1	Startwert Endwert	25 9,79	25 8,52	25 6	78720 25295	171 439,81	173
GradientZurückStartwert1	Startwert Endwert	25 9,56	25 8,51	25 6,12	78720 25195	171 439,99	304
GradientZurückStartwert2	Startwert Endwert	13 10,07	13 7,34	13 6,86	40934 25323	330 439,99	260
Vektorprojektion-Sigmakorrektur	Startwert Endwert	11 9,87	8 7,8	7 6,59	27072 25273	413,23 439,92	112
PluMiGlobal2	Startwert Endwert	11 7	8 25	7 1	27072 33887	413,23 320,2	350
PluMiGlobal	Startwert Endwert	11 5,75	8 17	7 1	27072 24391	413,23 439,79	3646

Abb. 4.5.2: Ergebniswerte

So wurde die Rechnung PluMiGlobal2 gestartet (Plus-Minus-Global); Und zwar mit relativ großen Schrittweiten, um einen Überblick über einen sehr großen Bereich zu erhalten. Auf diese Weise kamen mehrere recht interessante Werte zustande. Hier ist nur einer unter Endwert aufgeführt. Dieser ist nicht der beste Wert, sondern das ist ein Wert in der Nähe der anderen Ergebnisse. Aber der aufgezeigte Wert hat eine andere Konstellation der Dicken der Stäbe. So ist hier nicht die Variable 1 die größte, sondern mit Abstand die Variable 2. Der Ergebniswert ist zwar nicht besser als die schon ermittelten, aber er gibt Anlass, zu weiteren Untersuchungen. Aus diesem Grund wurden von diesem Punkt aus mehrere unterschiedliche Rechnungen gestartet. Dabei wurden mehrere Algorithmen ausgewählt, als auch die Schrittweiten und deren Verhältnisse geändert. Dazu muss man sagen, dass, wenn überhaupt, nur leichte Verbesserungen gegenüber dem Startwert erzielt werden konnten. Entweder die Algorithmen konnten das lokale Minimum nicht verlassen, oder, wenn sie doch konnten, dann gingen sie zu dem schon bekannten lokalen Minimum der ersten Rechnungen.

Alle diese Rechnungen befanden sich in dem angestrebten Bereich von bis zu ein paar hundert Rechnungen. Der nächste Rechenlauf sollte dies ändern. Plus-Minus-Global wurde erneut gestartet. Diesmal aber mit viel kleineren Schrittweiten, um die Umgebung des ersten lokalen Minimums zu untersuchen. Dieser Lauf wurde allerdings lange (12 h) laufen gelassen. In dieser Zeit schaffte er mehrere tausend Rechnungen und fand tatsächlich zwischen beiden Minima ein weiteres, das den besten Wert darstellt und das globale Minimum sein könnte.

Die Darstellung des Ergebnisplots der Spannungen von Plus-Minus-Global (**Abbildung 4.5.3**) zeigt, dass nicht alle Stäbe den gleichen Spannungswert haben. Der Unterschied ist in diesem Fall zwar nicht sehr groß, aber bei anderen Endwerten ist er doch größer (20 N/mm²) und es sind auch immer andere Stäbe, die den maximalen oder minimalen Wert aufweisen. Dieses Springen macht es den Algorithmen schwierig, Gradienten zu ermitteln, die wirklich eine Verbesserung bringen. Aus diesem Grund haben hier die Plus-Minus-Rechnungen, also solche mit ausgeprägt kombinatorischem Charakter die besten Ergebnisse erzielt.

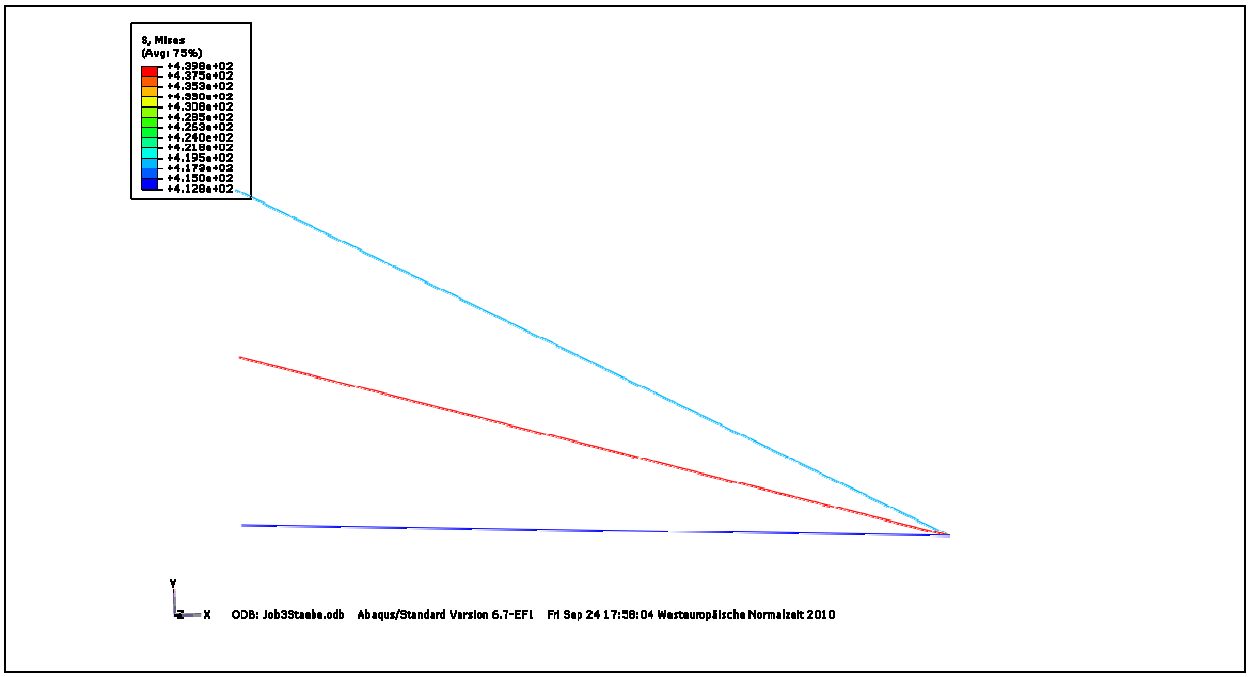


Abb. 4.5.3: Spannungsplot von PluMiGlobal

4.6 Das Modell „RePlatte3Var“

4.6.1 Der Modellaufbau

Die Platte ist 500 mm lang und am rechten Ende fest eingespannt. Am linken Ende ist sie mit einer Kraft belastet. Diese Kraft ist F_x = -10 kN und F_y = 10 kN. Das Modell ist als Shell konzipiert und mit Schalenelementen vernetzt (**Abbildung 4.6.1**). Das bedeutet, dass das Modell dreidimensional ist, vom geometrischen Aufbau jedoch zweidimensional. Die dritte Dimension wird durch Angabe einer Elementdicke erzeugt. Diese Technik kommt bei Modellen zum Einsatz, bei denen die Dicke sehr gering gegenüber den Ausdehnungen in die anderen Ebenen ist, z.B. bei Blechen. Von Seiten der Ergebniswerte her, muss man beachten, dass keine Änderungen in der dritten Richtung erfasst werden können. Die gewählten Elemente sind linear. Da ein Kraftanteil in y-Richtung zeigt, muss die Elementlänge so gewählt sein, dass bei

Erreichen der unteren Grenze Variablen, nicht nur eine Elementreihe entsteht, da diesen Ergebnissen, z.B. Spannungen, nicht getraut werden kann. Es müssen mindestens zwei Reihen gewährleistet sein.

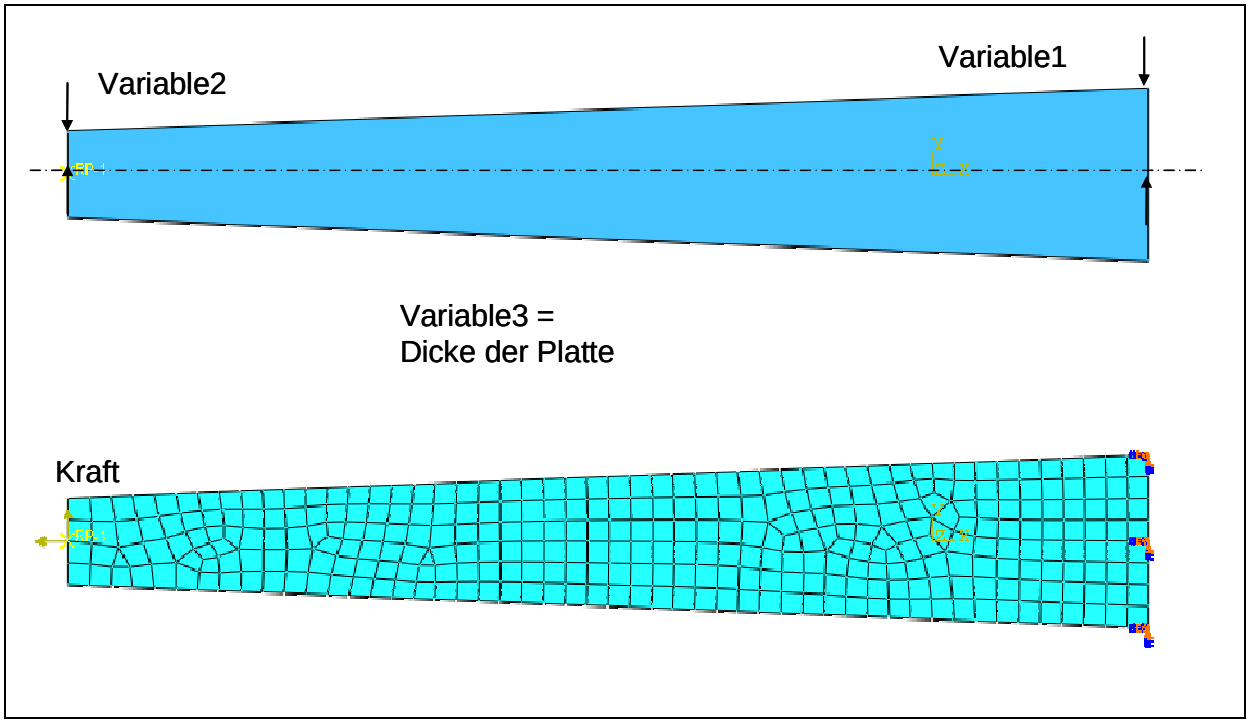


Abb. 4.6.1: Das Modell RePlatte3Var

Das Optimierungsproblem ist eine restringierte Gewichtsoptimierung. Die Variable 1 ist die halbe rechte Seitenlänge, Variable 2 die halbe linke Seitenlänge und die Variable 3 ist die Dicke der Platte. Durch die y-Komponente der Kraft F entsteht ein Moment, das vermuten lässt, dass die rechte Seitenlänge größer wird als die linke Länge. Den besten Volumenwert aus der Kombination der drei Variablen gilt es zu ermitteln.

4.6.2 Die Rechenläufe

Die Rechnungen mit Gradient-Zurück starten bei frei gewählten Werten, sind aber dabei im zulässigen Bereich und in einigem Abstand von der Sigmafläche (**Abbildung 4.6.2**). Diese Algorithmen müssen bei jeder einzelnen Variablenuntersuchung bessere Werte finden, um daraus Richtung und Schrittweite zu ermitteln. So kommt es, dass bei der ersten Rechnung ein Wert von 171,1 cm³ ermittelt wird. Wenn die zweite Rechnung auf die Sigmafläche trifft, sind die beiden Längenvariablen schon so weit reduziert, dass die Dicke nur noch auf einen kleinsten Wert von 8,72 mm kommt. Um die Dicke weiter zu reduzieren, müssten die Längen wieder größer werden, was nicht möglich ist. Deshalb ist nur ein schlechteres Ergebnis möglich.

		V1 [mm]	V2 [mm]	V3 [mm]	Ergebnis-wert [cm³]	Ergebnis [N/mm²]	Einzel-rechnun-gen
GradientZurück1	Startwert Endwert	100 72	100 26,2	6 3,5	600 171,1	145,34 438,72	138
GradientZurück2	Startwert Endwert	50 43,8	50 15,07	12 8,72	600 260,7	238,78 439.98	360
PlusMinus4	Startwert Endwert	100 130	100 15	6 1,25	600 112,5	145,34 407,31	302
PlusMinus5	Startwert Endwert	100 150	100 50	6 1,0	600 99,9	145,34 424,31	4
PluMiGlobal2	Startwert Endwert	70 170	70 40	6 1,0	- 105	- 427,84	3724
VektorSigmakorrektur	Startwert Endwert	84 217,74	20 77,64	3,5 0,5	182 73,8	414,84 425,91	110

Abb. 4.6.2: Tabelle der Rechenläufe

Die Rechnungen Plus-Minus haben weitere Fähigkeiten, mit denen noch bessere Werte erreicht werden können. Plus-Minus verschlechtert Variable 1, um gleichzeitig die Dicke sehr weit zu reduzieren. Plus-Minus 5 wurde dazu noch mit sehr großen Schrittweiten gestartet. Eher zufällig stößt er damit schon nach vier Rechnungen auf einen noch besseren Wert. Tendenziell zeigen beide Rechenläufe in die gleiche Richtung.

Zur Überprüfung dieser Ergebnisse wurde ein sehr langer Rechenlauf mit Plus-Minus-Global gestartet. Dieser sollte einen großen Bereich überprüfen und möglicherweise mehrere Alternativen zur Verfügung stellen. Dieser Lauf hat allerdings weder bessere Werte noch Alternativen gezeigt, sondern nur tendenziell die alten Ergebnisse bestätigt.

In einem gewissen Abstand zu diesem Optimum, wurde nun noch eine Rechnung mit Vektorprojektion-Sigmakorrektur gestartet. Durch diesen Abstand sollte die Möglichkeit gegeben werden, einen andern Weg zu gehen als die bisherigen Rechnungen, aber es zeigt sich, dass auch dieser Algorithmus das bisherige Optimum bestätigt. Er geht sogar noch weiter bis zu den Längen 218 mm und 78 mm. Dadurch kann die Dicke bis zu ihrer unteren Grenze verkleinert werden. An dieser Stelle lässt sich, durch die vielen Startwerte und Algorithmen gestützt, das globale Minimum vermuten(**Abbildung 4.6.3**).

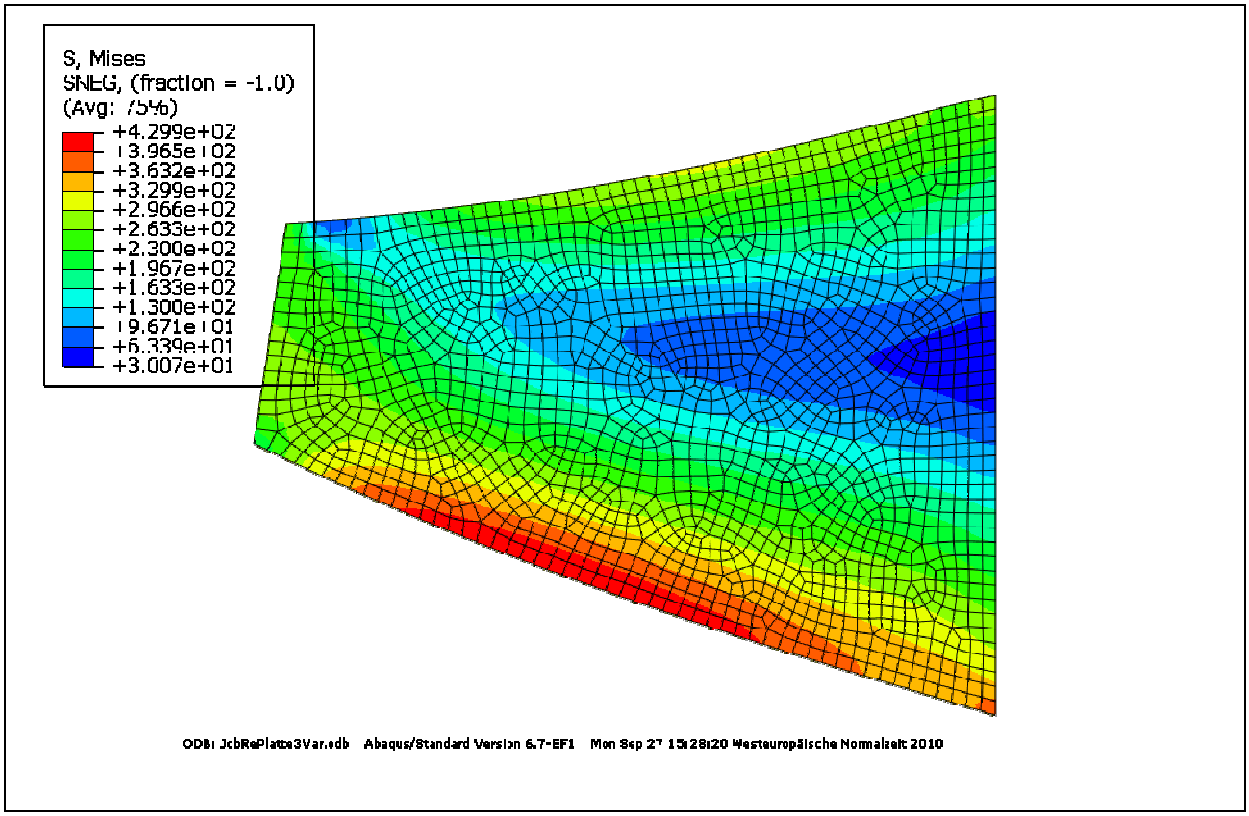


Abb. 4.6.3: Spannungsplot (Mises) von Vektorprojektion-Sigmakorrektur

Interessant wäre noch die Untersuchung, ob auch bei anderen Belastungen, z.B. ein Kraftanteil F_z, immer noch der unteren Rand der Dicke das Ergebnis wäre.

Bei diesem Modell ist die rechte Seite fest eingespannt, was zur Folge hat, dass ein prinzipieller Schwachpunkt aller FEM-Programme zum Tragen kommt. Angenommen, die Elementgröße würde immer weiter verringert, dann würden die am rechten Rand ermittelten Spannungen stetig wachsen. Da die Optimierung mit konstanten Elementgrößen rechnet, kann sie, wenn nicht die rechten Elemente die größten Spannungen aufweisen, korrekte Ergebnisse errechnen. Das Optimierungsprogramm bietet jedoch die Möglichkeit mit zwei Ergebnissets zu arbeiten. Das bedeutet, dass in der Modellerstellung der rechte Rand herauspartitioniert werden kann. So kann ein zweites Ergebnisset erstellt werden, in dem die Spannungen ermittelt werden, ohne die Spannungen der Elemente des rechten Rands. Dies wurde an diesem Modell durchgeführt. Wie in Abbildung 4.6.3 jedoch zu sehen ist, sind die größten Spannungen am unteren Rand. So ergab die zweite Methode sehr ähnliche Optimierungsergebnisse.

4.7 Das Modell „Kraftteilung4Var“

4.7.1 Der Modellaufbau

Das Modell Kraftteilung4Var besteht aus drei Rundstäben und einer Platte. Zwei Stäbe sind unverschieblich gelagert und ein Stab wird an seinem freien Ende mit einer Kraft oder einem Moment beaufschlagt. Modelliert wurde eine Beam-Struktur. Diese ist in **Abbildung 4.7.1** die untere Darstellung. Dort ist die Struktur zu sehen. Diesen Stäben werden Eigenschaften zugeordnet. So werden aus drei Stäben Rundstäbe mit einem gewissen Durchmesser und aus einem Stab wird eine Platte mit definierten Abmessungen. Im oberen Teil des Bildes wird dies grafisch dargestellt. Dort sind auch die Variablen des Modells zu sehen. Die Variable 1 verändert die Längen aller Stäbe gleichzeitig, sodass die Gesamtlänge des Bauteils immer bei 600 mm bleibt. Variable 2 verändert den Abstand der beiden rechten Stäbe und die Länge der Platte und Variable 3 ist der Durchmesser des linken Stabes und Variable 4 der Durchmesser der beiden rechten Stäbe. Für die FEM-Rechnung sind die Stäbe allerdings mit Beam-Elementen vernetzt, sodass sie vom mathematischen her runde Balken und keine Stäbe sind.

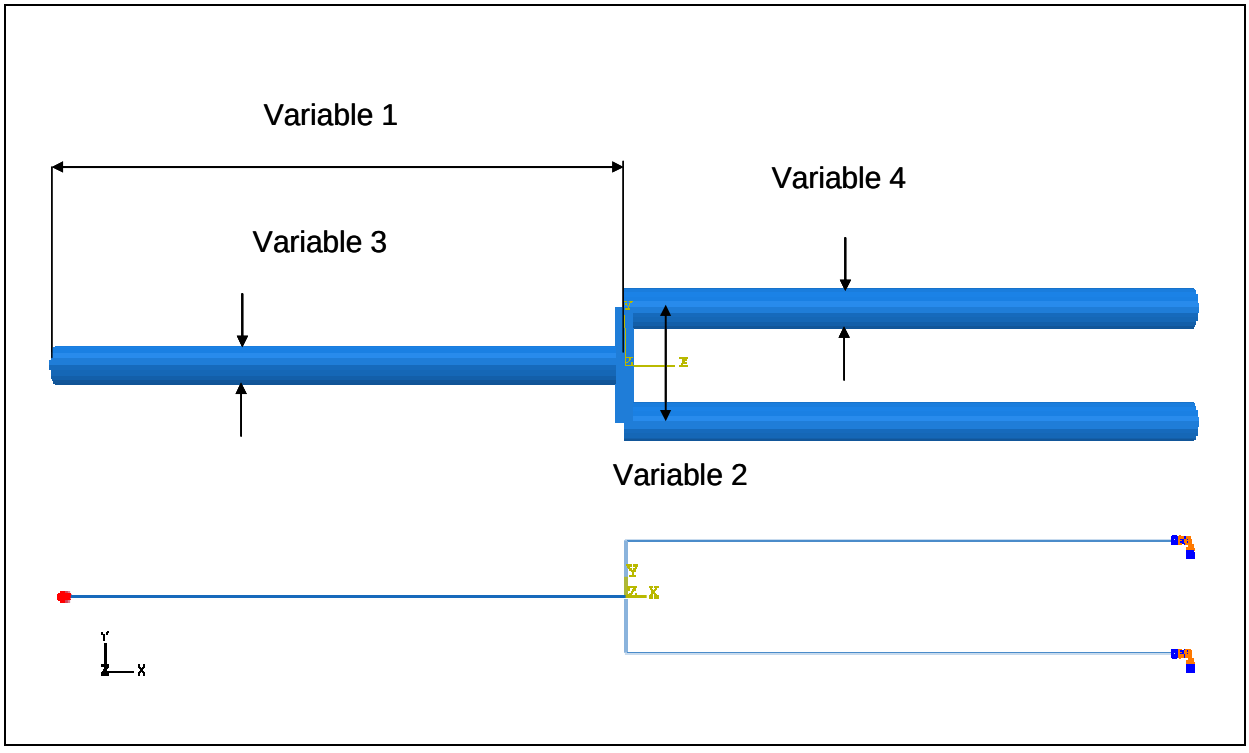


Abb. 4.7.1: Das Modell Kraftteilung4Var

Die beiden rechten Stäbe sind am rechten Ende fest eingespannt und der linke Stab ist an seinem linken Ende mit einem Moment belastet, das in die Blattebene hereinzeigt und 250 Nm groß ist.

Dieses ist die Erweiterung des zweidimensionalen Modells. Dort war der Zusammenhang die Optimierung betreffend der Variablen 2 und vier klar geworden. Welchen Einfluss die anderen beiden Variablen haben ist zu untersuchen.

4.7.2 Die Rechenläufe

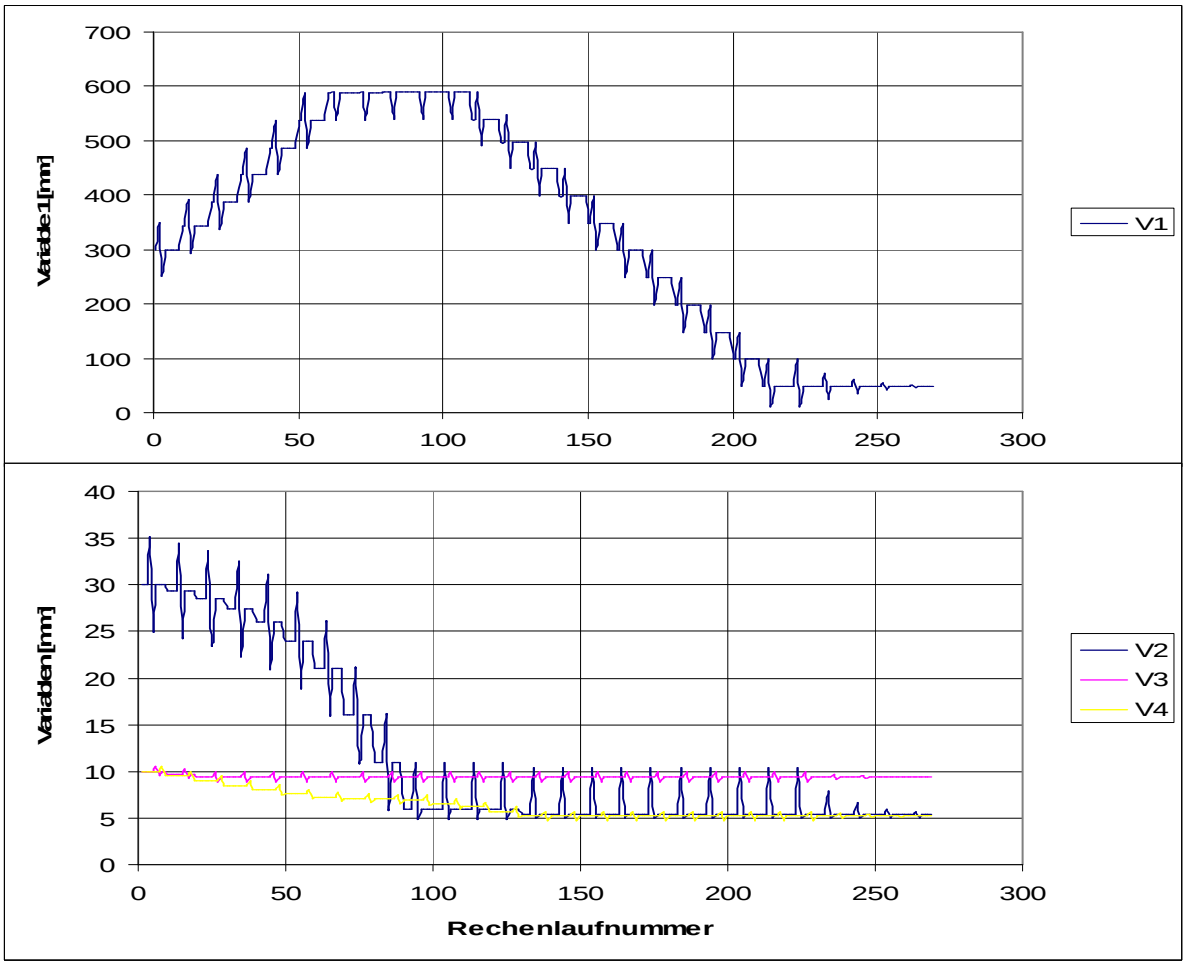


Abb. 4.7.2: Diagramm des Rechenlaufes Gradient-Zurück-2

In Abbildung 4.7.1 sind die Maße zu sehen, mit denen die Rechnung Gradient-Zurück-2 gestartet wurde. Die Variable 1 befindet sich in der Mitte ihres Definitionsbereiches (10 mm - 590 mm). Am Ende der Rechnung befindet sie sich an ihrem linken Ende. Bei Betrachtung des Ablaufs der Rechnung (**Abbildung 4.7.2**) stellt sich allerdings heraus, dass die Variable erst in die andere Richtung gegangen ist, und zwar bis an den anderen Rand. Dort verharrt sie auch, bis die Variablen 3 und 4 ihr Verhältnis zueinander so geändert haben, dass die Fläche der beiden linken Stäbe kleiner als die Fläche des einen rechten Stabes ist. Dann erhält die Variable 1 einen anderen Einfluss und sie geht in die andere Richtung, bis an den Rand des Definitionsbereiches. Das zeigt, dass die Variable 1 eine abhängige Variable ist. Sie ist von dem Verhältnis der zwei genannten Variablen abhängig, wodurch sich ihr Verlauf ergibt. Weiter ist festzustellen, dass der Einfluss der Radien

auf den Volumengradienten viel größer ist, als der der Längenvariablen. Das hat zur Folge, dass die echten Gradientenverfahren nur bedingt für dieses Problem geeignet sind. Sie können zwar unter bestimmten Voraussetzungen funktionieren, aber sie sind sehr langsam. Der Erfolg dieses ersten Rechenlaufes liegt daran, dass der von Gradient-Zurück berechnete Gradient ein unechter Gradient ist, das heißt, er ist nicht auf die Schrittlänge eins normiert, sondern auf die jeweiligen eingegebenen Schrittlängen. Die Schrittlänge der Variablen 1 war 50 mm, während die der Durchmesservariablen 0,5 mm war. Bei diesem Verhältnis lässt sich der Gradient so verbiegen, dass er auch an den Längen (Variable 1) entlanggeht.

Gut für dieses Problem geeignet, ist der Algorithmus Plus-Minus, da auch er die eingegebenen Schrittweiten als Maß nimmt. Bei gleichzeitiger Änderung der Variablen stellt er fest, wo sich etwas verbessert hat und nimmt diesen Wert als neuen Ausgangspunkt. Dabei ist anzumerken, dass auch Plus-Minus bei zu kurzer Schrittlänge der Variablen 1 an einem lokalen Minimum bei 300 mm hängen geblieben ist. Hier kam allerdings der Algorithmus Plus-Minus-Global zum Einsatz der die Schrittweiten so lange vergrößerte, bis das Minimum verlassen werden konnte. Die Rechenläufe Plus-Minus 4 und 5 ermitteln von unterschiedlichen Startpunkten aus das gleiche Minimum. Dieses wird als das Globale angesehen (**Abbildung 4.7.3**).

Weiter bleibt darauf hinzuweisen, dass die Variable 3, unabhängig von den andern Variablen, ihr Minimum bei ca. 9,3 mm hat. Dass liegt daran, dass das Modell mit einem Moment belastet ist, weshalb die Belastung des Stabes nicht von seiner Länge abhängig ist. Das heißt, das Problem könnte aufgeteilt werden. Ist die minimale Dicke des Stabes gefunden, könnte als dreidimensionales Problem weitergerechnet werden. Auch hat dieser Sachverhalt Auswirkungen auf die Gradientenermittlung, die im Abschnitt 4.7.3 behandelt wird.

		V1 [mm]	V2 [mm]	V3 [mm]	V4 [mm]	Ergebnis-wert [cm³]	Ergebnis [N/mm²]	Einzel-rech-nungen
GradientZurück2	Startwert Endwert	300 10	30 5,33	10 9,4	10 5,22	297,8 106,6	318,3 437,46	212
PlusMinus5	Startwert Endwert	10 10	5,33 25	9,4 9	5,22 2	106,6 29,9	437,46 436,64	108
PlusMinus4	Startwert Endwert	166 10	30 25	9,25 9,85	1,85 1,98	67,5 30	409,85 438.85	542

Abb. 4.7.3: Diagramm des Rechenlaufes Gradient-Zurück-2

In **Abbildung 4.7.4** sind die Ergebnisse von Plus-Minus-5 dargestellt. Zuerst sind die Maße zu sehen und folgend die Mises-Spannung im Optimum.

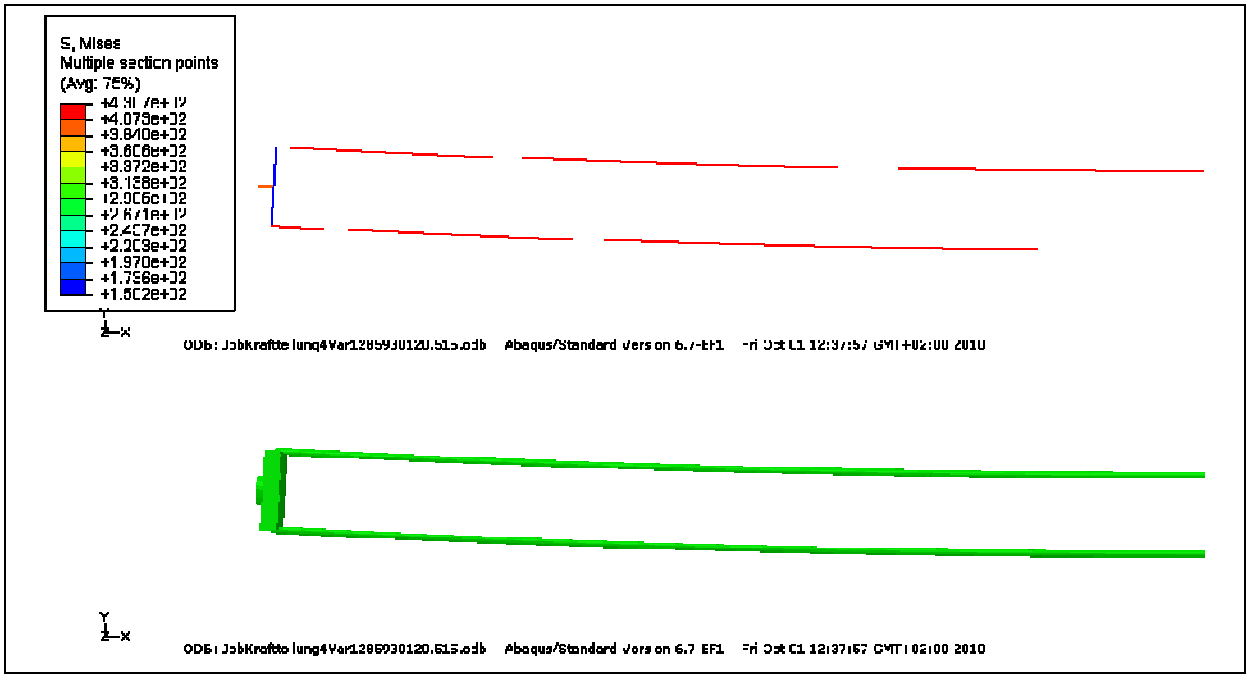


Abb. 4.7.4: Ergebnisplots von Plus-Minus-5

4.7.3 Das Modell und die Vektorprojektion

V_1	V_2	V_3	V_4	Ergebniswert [cm³}	Ergebnis (Mittelwert) [N/mm²]	
300	40	10	10	302,74	200,45	Start
310	40	10	10	299,602	209,21	Tast
300	45	10	10	305,24	199,89	Tast
300	40	11	10	322,53	162,95	Tast
300	40	10	11	342,33	197,7	Tast
299,9943785	39,9886044	10,07331913	9	268,31	202,04	Ausführung

Abb. 4.7.5. : Auszug aus der Ergebnisliste von Vektorpr.-Sigmak.-Mittelwert2

Neben den oben genannten Algorithmen ist auch ein Rechenlauf mit einem Gradientenverfahren mit Vektorprojektion gelaufen. Die ersten sechs Schritte dieses Rechengangs sind als Auszug abgebildet (**Abbildung 4.7.5**). Der erste Schritt ist als Startschritt gekennzeichnet. Es folgen vier Tastschritte und der daraus berechnete Ausführungsschritt. Die Berechnung des Schrittes erfolgt, wie bei allen Gradienten, die mit der Vektorprojektion arbeiten, nach den Gleichungen 3.2 bis 3.5. Für die Werte des Auszugs, die ersten Schritte des Rechenlaufs, wird sie beispielhaft durchgeführt.

Das Differenzenverfahren ergibt den Volumengradienten und den Spannungsgradienten. Da ein Gradient den stärksten Anstieg anzeigt, aber nach einem Minimum gesucht wird, wird der Volumengradient um 180 °gedreht.

$$\frac{299{,}6 - 302{,}74}{10} = -0{,}314 = -v_1 \; ; \qquad \frac{209{,}21 - 200{,}45}{10} = 0{,}876 = \sigma_1$$

$$\frac{305{,}24 - 302{,}74}{5} = 0{,}5 = -v_2 \; ; \qquad \frac{199.89 - 200{,}45}{5} = -0{,}114 = \sigma_2$$

$$\frac{322{,}53 - 302{,}74}{1} = 19{,}79 = -v_3 \; ; \qquad \frac{162{,}95 - 200{,}45}{1} = -37{,}5 = \sigma_3$$

$$\frac{342{,}33 - 302{,}74}{1} = 39{,}58 = -v_4 \; ; \qquad \frac{197{,}7 - 200{,}45}{1} = -2{,}75 = \sigma_4$$

$$\begin{pmatrix} +0{,}314 \\ -0{,}5 \\ -19{,}79 \\ -39{,}58 \end{pmatrix} = \vec{v} \qquad \begin{pmatrix} +0{,}876 \\ -0{,}114 \\ -37{,}5 \\ -2{,}75 \end{pmatrix} = \vec{\sigma}$$

Der negative Volumengradient wird auf den Spannungsgradienten projiziert.

$$\left(\frac{\begin{pmatrix} +0.876 \\ -0{,}114 \\ -37{,}5 \\ -2.75 \end{pmatrix} * \begin{pmatrix} +0{,}314 \\ -0{,}5 \\ -19{,}79 \\ -39{,}58 \end{pmatrix}}{(+0{,}876)^2 + (-0{,}114)^2 + (-37{,}5)^2 + (-2{,}75)^2} \right) * \begin{pmatrix} +0.876 \\ -0{,}114 \\ -37{,}5 \\ -2.75 \end{pmatrix} = 0{,}6 * \begin{pmatrix} +0.876 \\ -0{,}114 \\ -37{,}5 \\ -2.75 \end{pmatrix} = \begin{pmatrix} +0{,}526 \\ -0{,}078 \\ -22{,}5 \\ -1{,}65 \end{pmatrix} = \vec{b}_a$$

Der Ausführungsvektor ist der negative Volumengradient abzüglich der Vektorprojektion.

$$\vec{S} = \vec{v} - \vec{b}_a = \begin{pmatrix} +0{,}314 \\ -0{,}5 \\ -19{,}79 \\ -39{,}58 \end{pmatrix} - \begin{pmatrix} +0{,}526 \\ -0{,}078 \\ -22{,}5 \\ -1{,}65 \end{pmatrix} = \begin{pmatrix} -0{,}212 \\ -0{,}422 \\ +2{,}75 \\ -37{,}93 \end{pmatrix}$$

Dieser wird auf die einflussreichste Schrittlänge normiert.

$$\begin{pmatrix} -0{,}212 \\ -0{,}422 \\ +2{,}75 \\ -37{,}93 \end{pmatrix} :37{,}93*1= \begin{pmatrix} -0{,}0056 \\ -0{,}0011 \\ +0{,}073 \\ -1 \end{pmatrix} = \vec{S}'$$

Der normierte Ausführungsvektor wird zum alten Ausführungsschritt addiert. Das ist in diesem Fall der Startwert. Das ergibt den neuen Ausführungsschritt, die neuen Variablenwerte, die in der letzten Zeile des Auszugs stehen.

Der Ausführungsschritt ist orthogonal zum Sigma-Gradienten. Deshalb ist der Anteil von Variable 3 eher gering. Es macht sich aber auch der geringe Einfluss der Variablen 1 und 2 auf den Volumengradienten bemerkbar, weshalb auch deren Anteil am Ausführungsschritt sehr gering ist. Den Haupanteil hat Variable 4. Bei Betrachtung des gesamten Rechenlaufs ändert sich das Bild in Bezug auf die beiden ersten Variablen nicht. Der Einfluss von Variable 3 ist ähnlich der Variable 4. Variable 1 und Variable 2 erreichen aber laufend keine großen Werte, weshalb der Algorithmus Vektorprojektion für dieses Modell nicht geeignet ist.

Bei dieser Rechnung wurden die gemittelten Spannungswerte des gesamten Modells genutzt und nicht, wie sonst, der maximale Spannungswert. Dieser Wert ist bei einem Wert der Variablen 3 nahe 9,5 mm immer in dem linken Stab. Er ändert sich nicht bei Änderung der anderen Variablen, sodass die Zeilen 1, 2 und 4 des Spannungsgradienten gleich null wären. Dies lässt sich nur mit den gemittelten Werten verhindern.

An diesem Modell ist die Variable 1 etwas Besonderes. Dies ist eine abgeleitete oder abhängige Variable, da sie kein eigenes Volumen hat, das sie ändern kann, sondern nur die Volumen von Variable 3 und 4. Diese beiden können z.B. in einem solchen Verhältnis stehen, dass die Änderung der Variablen 1 kein Volumen ändert. Auch kann es sein, dass der Gradient von 2 durch die gleichzeitige Änderung der Variablen 3 und 4 und deren Körper nicht mehr stimmt. Dies ist für die Anwendung eines Gradientenverfahrens bei diesem Modell problematisch.

4.8 Das Modell „Pleuel“

4.8.1 Der Modellaufbau

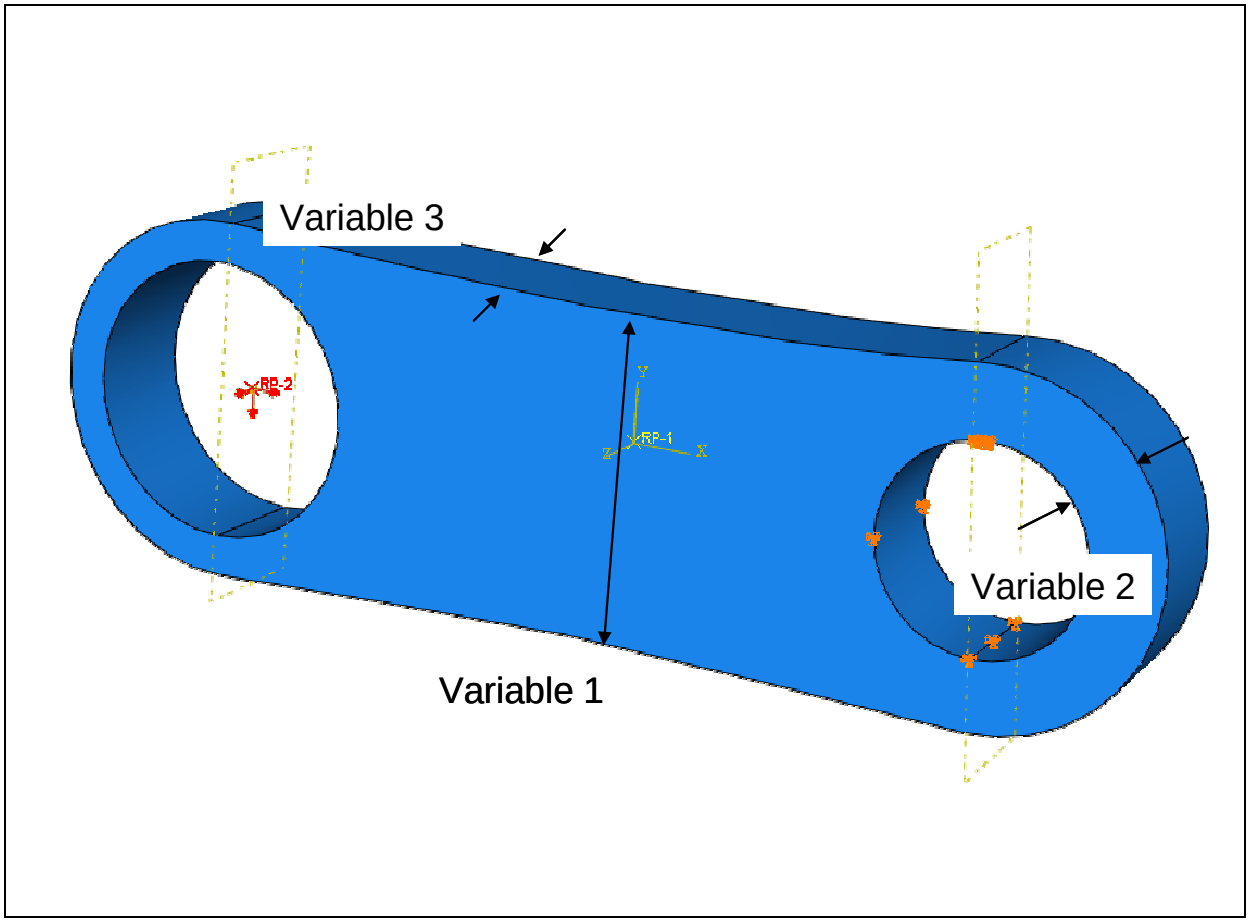


Abb. 4.8.1: Das Modell Pleuel

Die Größe der Variable 1 beträgt in dieser Darstellung 80 mm und die Länge des Pleuels beträgt 290 mm. Das Modell ist je nach der Art der Belastung eher als universeller Kraftübertrager als spezielles Pleuel zu betrachten. Der Name kommt eher aus der körperlichen Ähnlichkeit als von der spezifischen Belastung (**Abbildung 4.8.1**).

Die rechte Seite ist an der Aussparung fest eingespannt, die linke Seite ist an ihrer Aussparung durch eine Kraft belastet. Die Kraft beträgt (**Abbildung 4.8.2**):

F_x	400 kN
F_y	-5 kN
F_z	30 N

Abb. 4.8.2: Belastungen

Das Modell Pleuel ist dreidimensional konzipiert und mit Volumenelementen vernetzt. Dadurch können Änderungen der Spannungen in der z-Achsenrichtung berechnet werden, was bei Schalenelementen nicht möglich ist. Drei Variablen definieren das Optimierungsproblem, das eine Gewichtsoptimierung mit Spannungsrestriktion ist.

4.8.2 Die Rechenläufe

		V1 [mm]	V2 [mm]	V3 [mm]	Ergebnis-wert [cm³]	Ergebnis [N/mm²]	Einzel-rechnun-gen
GradientZurück	Startwert Endwert	70 54,89	45 35	25 19,25	410,44 244,09	295,2 451,57	100
PlusMinus	Startwert Endwert	40 60	41,62 35	28 18,5	356,9 244,1	455,25 450,19	144
VektorSigmakorrektur	Startwert Endwert	40 54,92	41,62 35	28 18,97	356,9 240,56	455,25 456,99	634

Abb. 4.8.3: Das Modell Pleuel

Der Algorithmus Gradient-Zurück wurde an für ihn geeigneten Startwerten gestartet (**Abbildung 4.8.3**). Die maximale Spannung der nullten Rechnung ist 295,2 N/mm². Die Werte weisen also einen Abstand zur Spannungsfläche auf, was für diesen Algorithmus notwendig ist, da er Variablen nur verbessern kann. So sind die Ergebniswerte der Variablen auch sämtlich kleiner als die Startwerte.

Anders ist es bei den Algorithmen Plus-Minus und Vektorprojektion-Sigmakorrektur. Sie starten mit Variablenwerten, deren Ergebnisse sehr nah an der Spannungsfläche liegen. Aber sie können Werte der Abmessungen vergrößern, um im Gegenzug die Dicke so zu verkleinern, dass am Ende das gleiche Optimum erreicht wird, das auch Gradient-Zurück erreicht hat. Im Vergleich sieht man den Nachteil von Plus-Minus. Er hat nicht die Möglichkeit der Schrittweitenminimierung. Deshalb findet Vektorprojektion-Sigmakorrektur die besten Werte. Bei diesen Ergebnissen ist zu erwähnen, das die untere definierte Grenze der Variablen 2 = 35 mm ist. Diese wurde schon früh im Rechenlauf erreicht. Trotzdem läuft der Algorithmus sauber weiter. Das Erreichen der Randbedingung kann zu einem Problem werden, da der ermittelte Ausführungsschritt dann nicht den Gradienten entsprechend verändert wird. Dies ist als mögliche Problemursache zu bewerten.

Nicht aufgeführt in der Darstellung ist der Rechengang Penalty-Vektorprojektion. Er verhält sich wie in vorherigen Untersuchungen festgestellt. Er läuft im nichtzulässigen Bereich, da er nicht die Möglichkeit hat, die Konvexität des Verlaufes der Spannungslinien auszugleichen. Trotzdem hat er einen gewissen Vorteil. Er ist sehr

zügig, da er nicht die langwierigen Korrekturschritte durchführt, hat aber prinzipiell den ähnlichen Verlauf wie Vektorprojektion-Sigmakorrektur. Wenn der Verlauf auf der Stelle tritt oder keine Erfolge der Volumenminimierung bringt, kann auf den Algorithmus Penalty-Gradient-Linear gewechselt werden. Dieser hat die Möglichkeit, in einem gewissen Maße den unzulässigen Bereich zu verlassen und das Minimum zu erreichen. Dies kann eine Alternative zu den beiden genannten Algorithmen sein.

Das Ergebnis der Variablen des Laufes Vektorprojektion-Sigmakorrektur und die endgültigen Spannungen sind in **Abbildung 4.8.4** dargestellt.

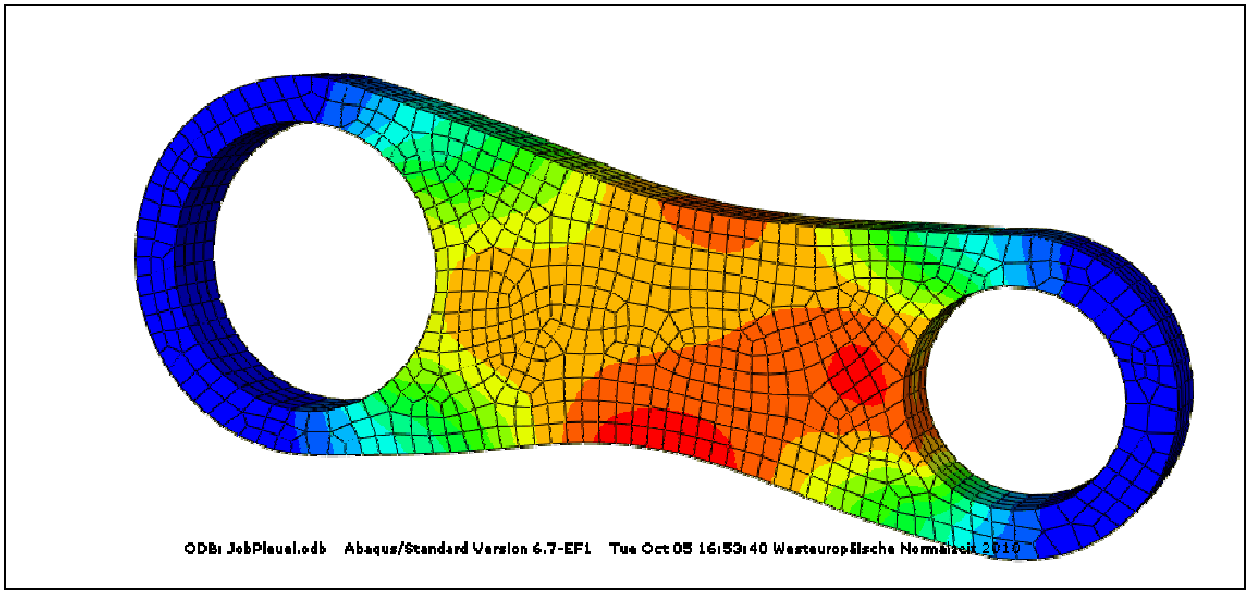


Abb. 4.8.4: Ergebnis des Modells Pleuel (Mises-Spannungen)

4.8.3 Zweite Version des Modells Pleuel

Eine zweite Version des Modells bekam mehrere Variablen für die Kontur, die das Pleuel beschreiben. Dadurch war es möglich, das Material besser auszunutzen, was durch den höheren Anteil an hochbelastetem Material (rot) zu sehen ist. Allerdings ist anzumerken, dass die beiden Modelle nicht zu hundert Prozent zu vergleichen sind, da die zweite Version anders belastet ist. Sie ist lediglich durch eine Druckkraft in x-Richtung belastet, trotzdem ist die Beobachtung der besseren Materialausnutzung tendenziell richtig.

Das Modell wurde mit mehreren unterschiedlichen Algorithmen behandelt, die aber zu ähnlichen Ergebnissen kamen. So kamen Halbe-Schrittweite, Gradient und Gradient-Nebenbedingung zu sehr ähnlichen Ergebnissen. Lediglich Goldener-Schnitt konnte an diese Ergebnisse nicht herankommen (**Abbildung 4.8.5**).

	Ergebniswert [cm³]	Ergebnis [N/mm²]	Einzel-rechnungen
Halbe Schrittweite	756,34	63,7	878
Goldener Schnitt	869,07	63,94	337
Gradient	759,21	63,95	279

Abb. 4.8.5: Die zweite Version des Modells Pleuel und die Ergebnisse

Weiterhin wurden zwei Rechnungen mit Vektorprojektion-Sigmakorrektur gestartet. Beide Startwerte waren schon in der Nähe des Optimums, konnten aber eindeutig über brauchbare Ausführungs- und Korrekturschritte die Ergebniswerte verbessern.

Die Variablen des Modells sind in **Abbildung 4.8.6** markiert aber nicht nummeriert. Der ersten Version entsprechen läuft die Nummerierung links beginnend nach rechts und die letzte Variable ist die Dicke des Pleuels.

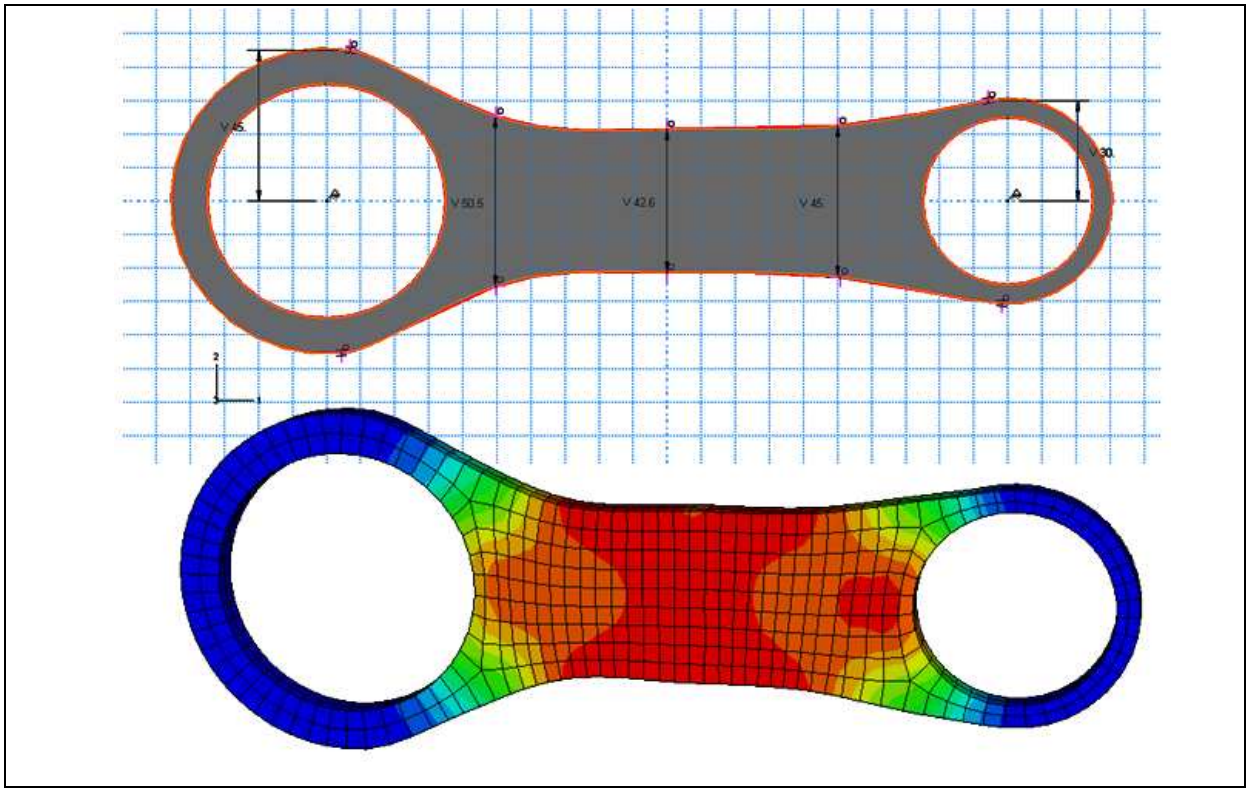


Bild 4.8.6: Die zweite Version des Modells Pleuel und das Ergebnis Gradient

4.9 Das Modell „Momentenplatten“

4.9.1 Der Modellaufbau

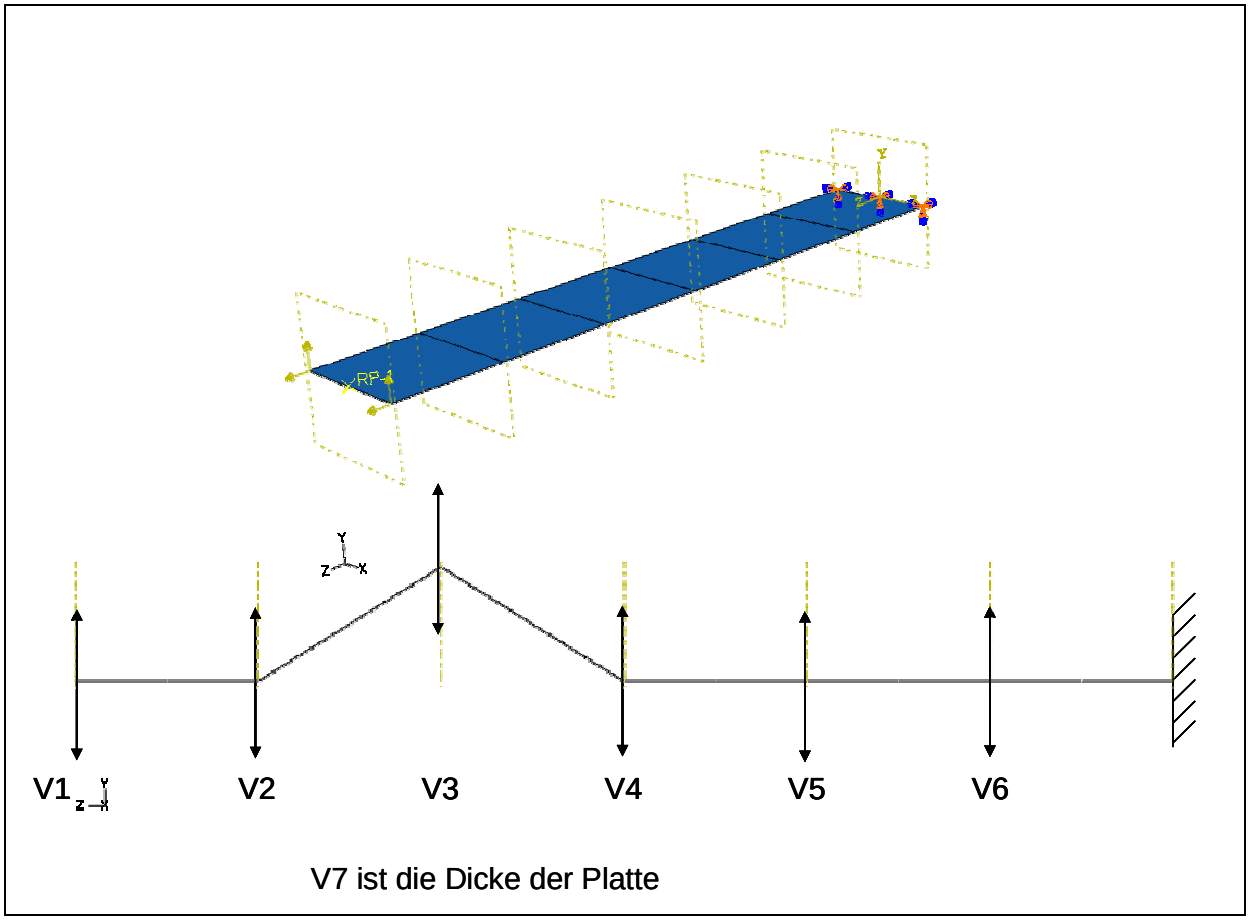


Abb. 4.9.1: Modell Momentenplatten

Das Modell ist die Weiterentwicklung des Modells „Momentenplatte2Var“. Die neue Platte ist allerdings in 6 Segmente unterteilt. Außerdem hat sie eine Breite von 160 mm. Sie hat eine Länge von 960 mm und eine Dicke von einigen Millimetern. Sie ist an ihrem linken Ende mit einer Kraft (F_y = 200 N, F_z = 400 N) belastet und am rechten Ende fest eingespannt. Die Platte besteht aus sechs Segmenten, die an ihren jeweiligen Enden in y-Richtung verschiebbar sind. Diese Enden sind, von links beginnend, die sechs Variablen. In **Abbildung 4.9.1** sind demnach die Variablen 1 bis 6 dargestellt. Die siebte Variable ist die Dicke der Platte. In der Nulllage der Platte haben die Variablen eins bis vier den Wert 500 mm und die Variablen fünf und sechs den Wert 300 mm. Die Variable 3 hat in dieser Darstellung den Wert 600 mm. Das Modell ist eine Platte, die mit Schalen-Elementen vernetzt ist.

Die konstruktive Ausgangslage ist die waagerechte Platte, die ein sehr großes Moment aufnimmt. Wird diese einfach dickenoptimiert, erhält man eine Dicke von 5,6875 mm und ein Volumen von 873,6 cm³. Bei einem Vergleich mit den formoptimierten Platten der folgenden Rechnungen sieht man, dass die Annahme

stimmt, dass man bei einer gezielten Drehung der Platte diese zwar verlängert, gleichzeitig aber das wirkende Moment verkleinert, sodass die Dicke soweit verkleinert wird, bis das resultierende Volumen optimiert ist.

Es ist zu beachten, dass die Dickenoptimierung nicht als einzige Variable die Dicke hatte, sondern auch die Formvariablen. Allerdings waren diese „auf Null gesetzt“. Das heißt, die Startformation war die wagerechte Platte. Von dieser Konstellation ausgehend, wird jede Formveränderung eine Verschlechterung des Volumenwertes nach sich ziehen. Als Folge wird nur die Dicke variiert. Von diesem Startwert aus muss also einer der weiterentwickelten Algorithmen genutzt werden.

4.9.2 Die Rechenläufe

Erste Rechnungen wurden mit den Algorithmen Halbe Schrittweite, Gradient, Gradient-Lang und Gradient-Nebenbedingung durchgeführt. Dabei wurden unterschiedliche Startwerte untersucht. In **Abbildung 4.9.2** ist eine völlig zufällig gewählte Startkombination zu sehen und das sich daraus ergebende Ergebnis (**Abbildung 4.9.3**). Dazu ist zu sagen, dass die jeweiligen Algorithmen bei gleichen Startwerten und gleichen Schrittlängen sehr ähnliche Ergebnisse ermittelten. So kann man feststellen, dass die genannten Algorithmen mit diesem Modell Probleme haben. Zwar ermitteln sie sauber nachvollziehbare Ergebnisse, aber sie sind nur in der Lage das Modell zu „strecken“ und nicht zu drehen. So wird eine gerade Linie aus der anfangs unregelmäßigen Form, aber diese Linie müsste noch um den Anbindungspunkt gedreht werden, können, um theoretisch das Optimum erreichen zu können. Denn, wie schon im entsprechenden zweidimensionalen Modell Momentenplatte erforscht, befindet sich das Optimum wahrscheinlich dort, wo das Moment verschwindet (oder zumindest in der Nähe davon). Das wäre eine gerade Linie, bei der die Variable 1 den Wert 980 mm hat.

Sollte das Ergebnis eine gerade Linie sein, wäre das eine Erkenntnis die erst gewonnen werden müsste. Im Nachhinein wäre diese siebendimensionale Version der Momentenplatte eher ein Fall für andere Belastungen, z. B. für eine Flächenlast. Aus didaktischen Gründen der Überprüfbarkeit und zum Nachvollziehen des Ergebnisses, macht diese Wahl der Belastung jedoch einen Sinn.

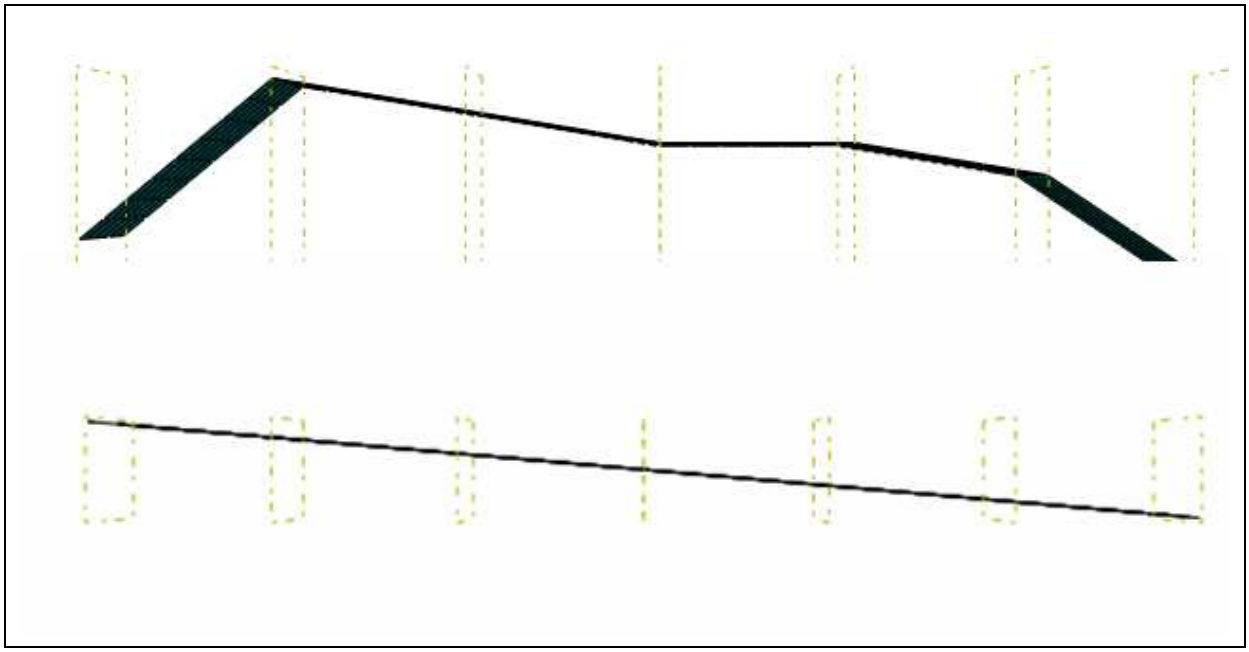


Abb. 4.9.2: MomentenplattenStartEnd1Grades: oben Start unten End

	Gradient Lang4GS	**Gradient**	**Gradient Neben**
Volumen [mm³]	797094	808595	808598
Mises [N/mm²]	439,84	439,91	439,84
Dicke [mm]	5,17	5,25	5,25
Nummer	756	1431	1011

Abb. 4.9.3: Tabelle von Momentenplatten

4.9.3 Die Rechenläufe Vektorprojektion-Sigmakorrektur und Plus-Minus

Vek.-Sig	Startwerte	Endwerte
Volumen [cm³]	808,6	521,3
Mises [N/mm²]	439,77	439,94
Dicke [mm]	5,25	2,99
Nummer	1	532

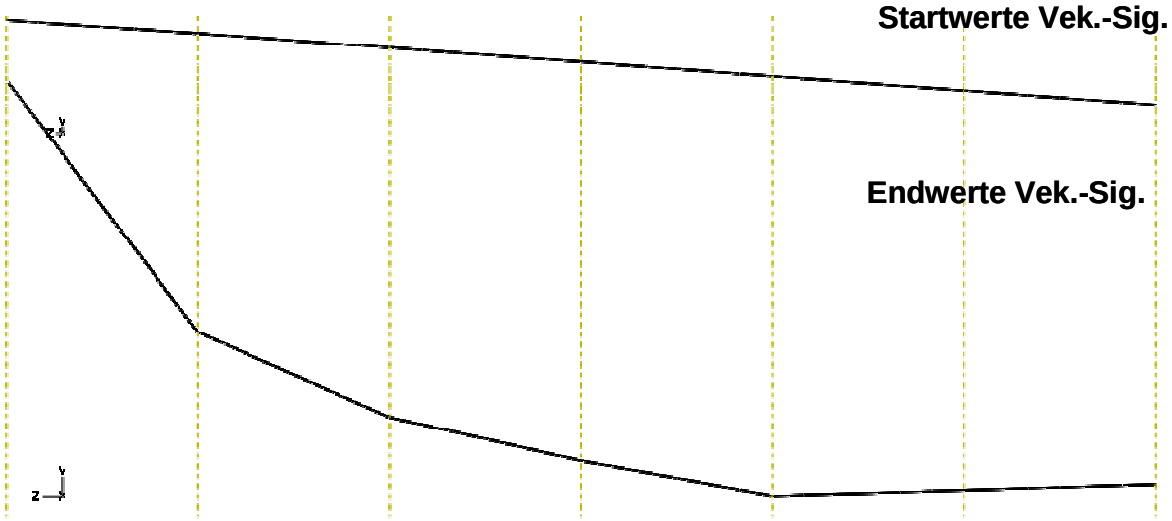


Abb. 4.9.4: Vektorprojektion-Sigmakorrektur, Startwerte und Endwerte

Die Optimumsermittlung wurde mit den Algorithmen Vektorprojektion-Sigmakorrektur und Plus-Minus fortgeführt (**Abbildung 4.9.4 und 4.9.5**). Die Rechnungen wurden mit Endwerten der vorherigen Rechnungen gestartet. Mit Vektorprojektion-Sigmakorrektur konnten die Ergebnisse noch einmal erheblich verbessert werden, da er das schon erwähnte „Drehen“ um den Aufhängungspunkt beherrscht, so dass das wirkende Moment so klein wurde, dass die Dicke des Bauteils noch einmal erheblich reduziert werden konnte.

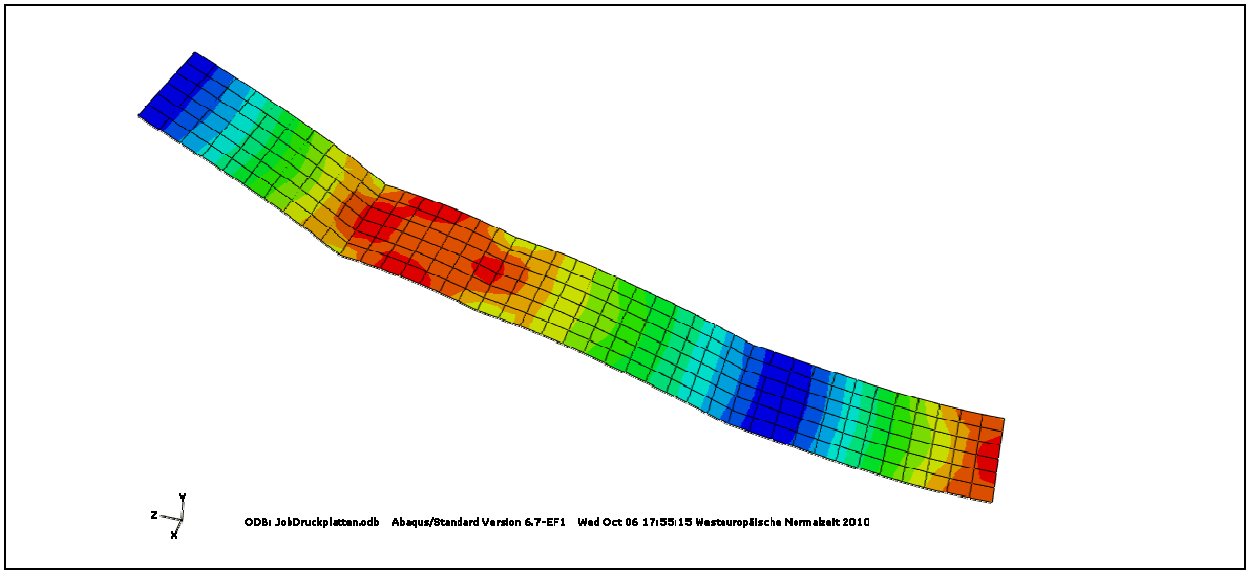


Abb. 4.9.5: Vektorprojektion-Sigmakorrektur,.Spannungsplot

Eine Verbesserung konnte also festgestellt werden. Dass sich aber diese Bogenform herausgebildet hat, hat den folgenden Grund: Den mit Abstand größten Einfluss auf die Spannung und das Volumen hat die Dicke der Platte. Hat nun die andere Variable auf beides nur einen geringen Einfluss, kann der in **Abbildung 4.9.6** gezeigte Fall eintreten und die einflusslose Variable bekommt einen negativen Wert. Nun könnte die Spannungslinie im Ursprung des Koordinatensystems solange gedreht werden, bis der Ausführungsschritt (roter Pfeil) für beide Variablen positiv wird. Diese Drehung würde durch eine Änderung des Verhältnisses der Spannungsänderungen geschehen. Und genau das ist der Fall in diesem Beispiel. Der Einfluss der Variablen zwei bis sechs auf die Spannung ist häufig so gering, das sie nur sehr kleine Anteile am Ausführungsschritt haben und diese sind dann auch noch häufig negativ. Dies trifft auf die erste Variable (linkes Ende) und die siebte Variable (Dicke) meistens nicht zu. Die Dicke wurde schon beschrieben (großer Einfluss auf beide Gradienten) und die erste Variable hat meistens einen ausreichenden Einfluss auf die Spannung, so dass sie im Ausführungsschritt einen ausreichenden Anteil erhält. Auch wenn das Ergebnis der Rechnung absehbar nicht optimal sein wird, so ist doch jeder einzelne ermittelte Gradient korrekt.

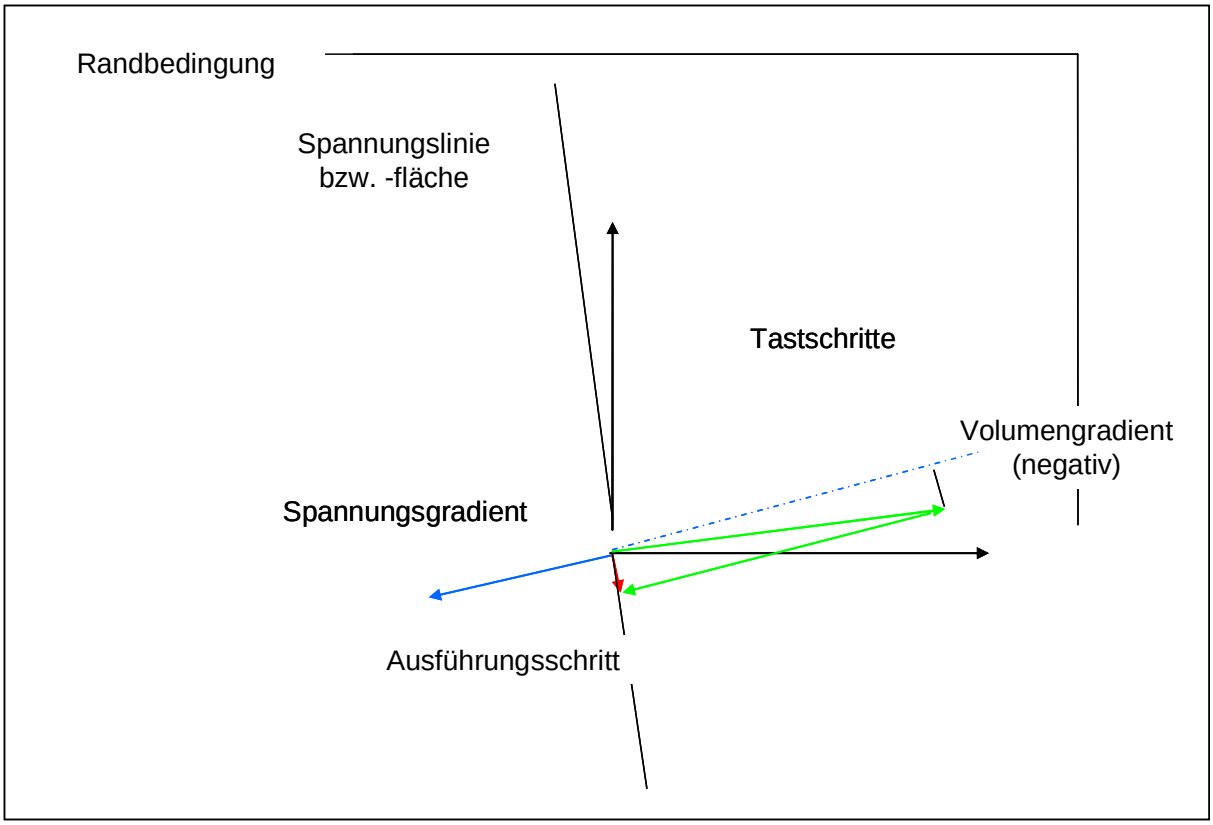


Abb. 4.9.6: Vektorprojektion

Diese Probleme mit dem Modell hat der Algorithmus Plus-Minus nicht. Er schafft es die Drehung zu erzeugen und auch die Streckung. So wird eine relativ genaue Gerade (leichte „Treppenbildung“) bei einem Wert der ersten Variable im Bereich von 1000mm errechnet. Die Verbesserung zu „Vek.-Sig.“ beträgt knapp einhundert Prozent. Der Nachteil an diesem Verfahren ist allerdings wiederum die hohe Rechendauer. Einige Stunden waren für die 2000 Rechnungen notwendig. Eine Möglichkeit wäre noch gewesen, das Ergebnis von Nummer 770 zu nutzen und von dort mit einem Gradientenverfahren weiterzumachen und die sehr stufige Form in recht kurzer Zeit zu strecken.

4.9.4 Das Modell Momentenplatten mit Segmenten

Um die Optimierungsmöglichkeiten der Platte zu steigern, wurde sie mit einzelnen Segmentdicken erweitert, so dass die einzelnen Bereiche ihre jeweilige optimierte Dicke erhalten kann (**Abbildung 4.9.7 und 4.9.8**). Dieses Problem wurde nur mit Gradient und Gradient-Lang bearbeitet. Beide kamen auf ein ähnliches Ergebnis. Heraus kam eine Dickenverteilung, die erwartungsgemäß zum freien Ende hin abnimmt. Für jedes Segment lohnt es sich, durch Verlängerung des Segments das Moment abnehmen zu lassen in einem größeren Maße, als die Spannung durch Dicke zu verringern, so dass jedes Mal das neue Segment etwas steiler eingestellt

wird. Dadurch bildet sich eine Kurve. Ansonsten gilt wie für die andere Version von Momentenplatten, dass mit diesen Algorithmen keine Drehung möglich ist.

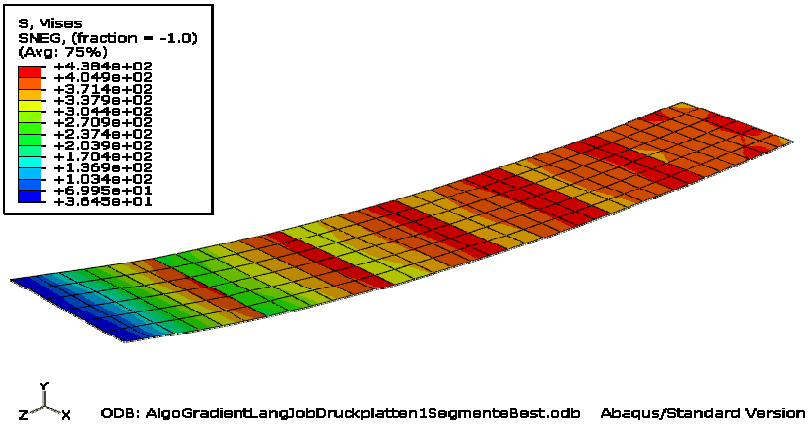
	Gradient Lang	Gradient
Volumen [mm³]	588125	612395
Mises [N/mm²]	439,99	439,98
Nummer	1581	1202

GradientLang Endwerte:

V1=307 V7=5,3
V2=315 V8=4,9
V3=524 V9=4,2
V4=533 V10=3,6
V5=545 V11=2,9
V6=563 V12=1,9

Abb.4.9.7: Momentenplatten Segmente Ergebnisse

Mit Segmenten



Ohne Segmente

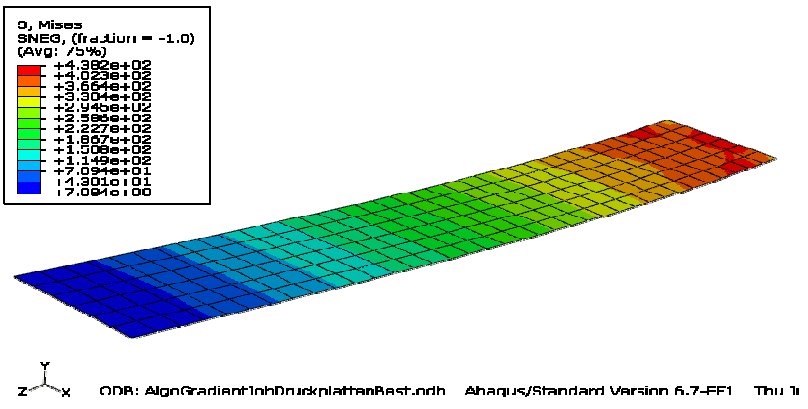


Abb.4.9.8: Momentenplatten Segmente Ergebnisse

4.10 Das Modell Stabtragwerk

4.10.1 Der Modellaufbau

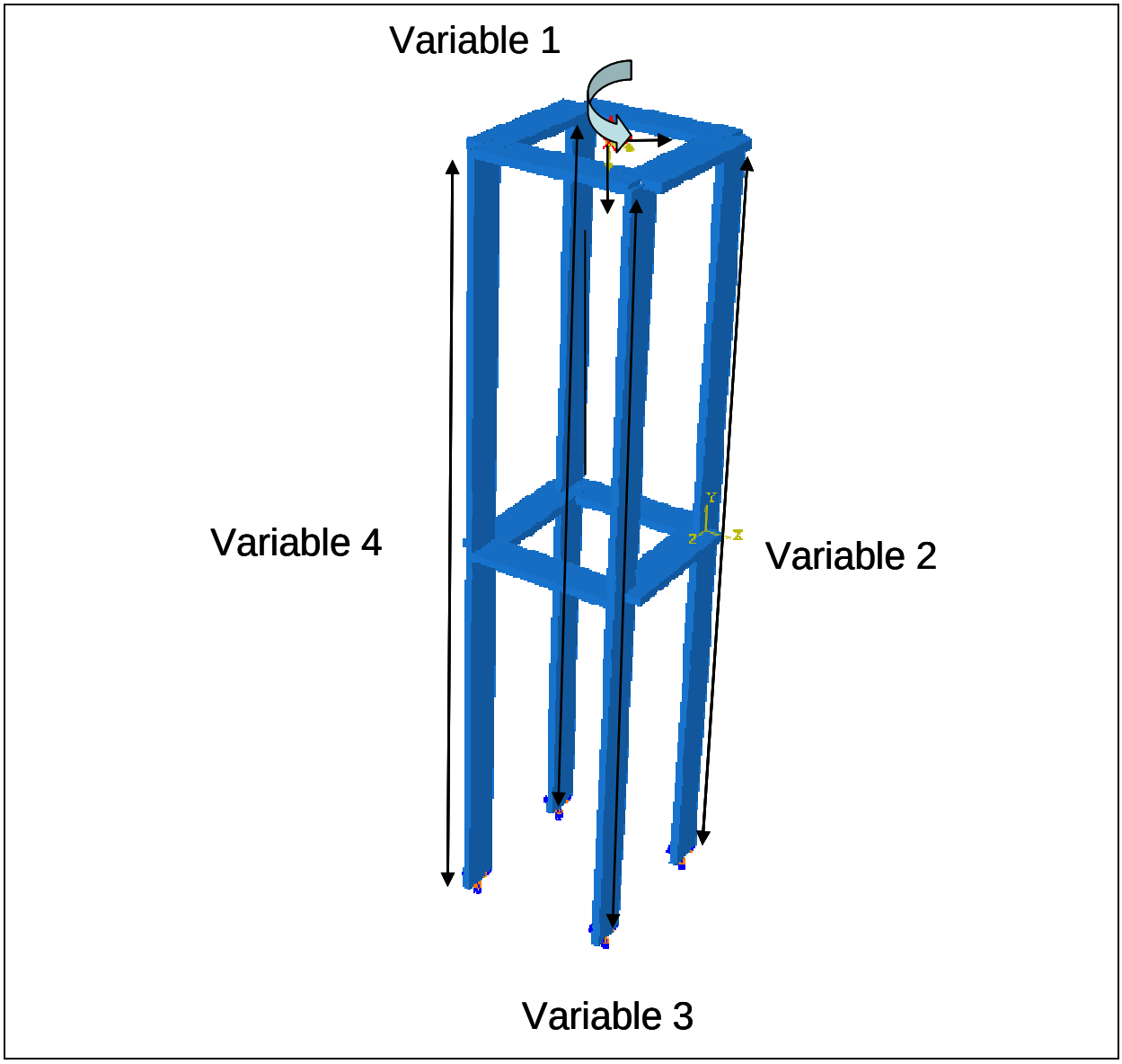


Abb. 4.10.1: Modell

Die meisten in der Strukturoptimierung behandelten Probleme sind Gewichtsoptimierungen, wie auch in dieser Arbeit. Trotzdem soll noch geschildert werden, dass dieses Optimierungsprogramm auch noch weitere Fähigkeiten aufweist. So kann sie auch Steifigkeiten eines Bauteils optimieren.

Im folgenden Beispiel, dem Modell Stabtragwerk wird diese Optimierungsart durchgeführt. Es besteht aus mehreren Blechen mit den breiten 50 und 10 mm. Das Modell ist einen Meter hoch und mit der Erde fest verbunden. Die mittleren Bleche sind an ihren Anbindungspunkten an den senkrechten Blechen verschiebbar, wodurch 4 Variablen entstanden sind. Nur bei Variable vier ist exemplarisch ein Pfeil

für den Freiheitsgrad eingetragen. Das gilt so auch für die anderen Verbindungspunkte und Variablen (**Abbildung 4.10.1**).

Im oberen Zentrum des Modells befindet sich ein Referenzpunkt, der über einen Coupling Constraint mit den obersten Blechen verbunden ist. An diesem werden unterschiedliche Kräfte und Momente eingeleitet. Es existieren die Versionen eins und zwei. Version eins ist im oberen Bild eingezeichnet und hat als Krafteinleitung ein Moment um die Hochachse von 10 Nm, einen senkrechten Kraftanteil von 10 kN und einen waagerechten Anteil in x-Richtung von 10 kN. Die zweite Version hat zwei waagerechte Anteile in x- und z-Richtung von 10 kN.

4.10.2 Die Rechenläufe

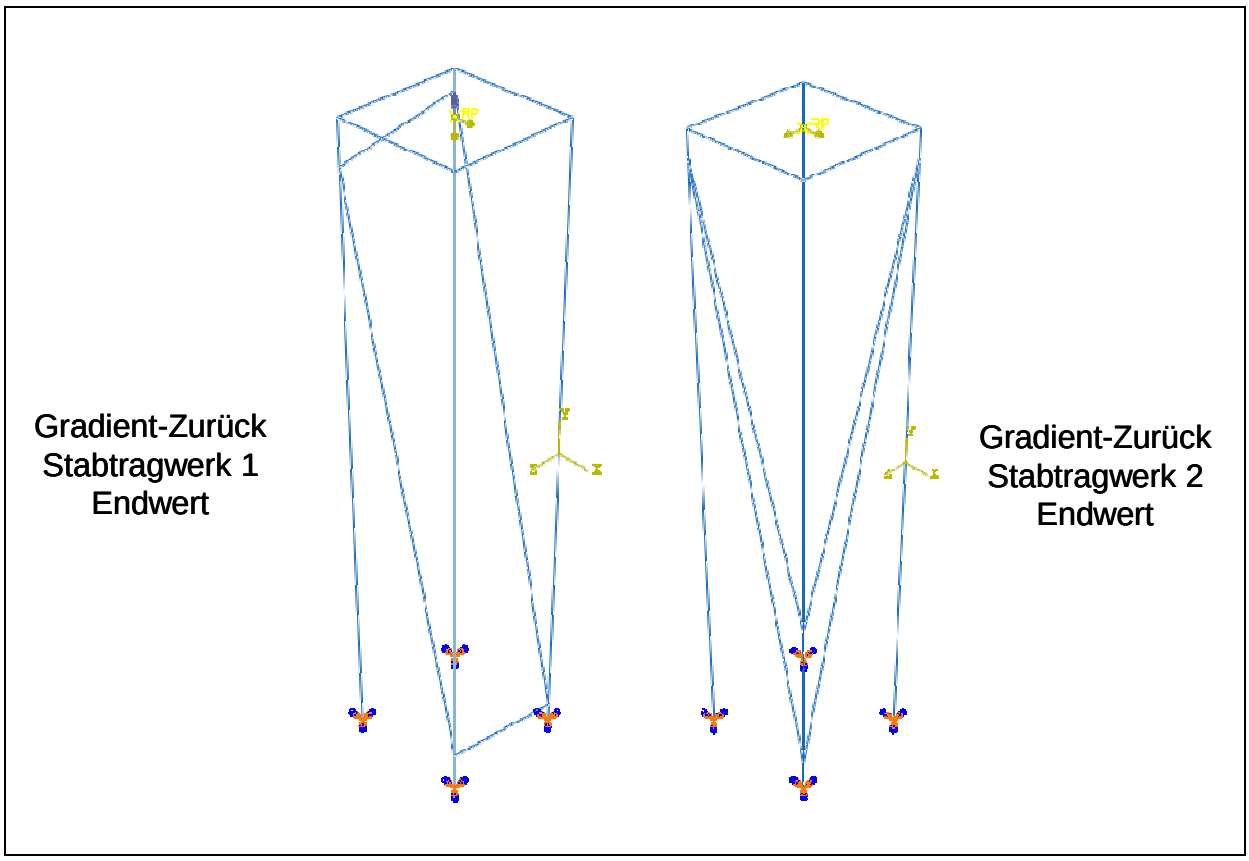


Abb. 4.10.2: Ergebnisse

Zur Belastungsversion zwei wurden drei Simulationsläufe durchgeführt. Zwei Rechenläufe wurden mit Gradient-Zurück und Plus-Minus gestartet. Beide hatten die gleichen Startwerte, doch ermittelten beide unterschiedliche Konstellationen. Da diese nicht unerheblich voneinander unterschieden, und Plus-Minus durch mangelnde Schrittweitenhalbierung sich dem Minimum nicht beliebig annähern kann, wurde in der Plus-Minus Konstellation ein weiterer Gradient-Zurück-Lauf gestartet. Das Ergebnis daraus ist, das die von Gradient-Zurück ursprünglich ermittelte

Konstellation die optimale ist. Da zwei sehr unterschiedliche Lösungsmöglichkeiten untersucht und verglichen wurden, bestätigt dies den Wert des ermittelten Optimums (**Abbildung 4.10.2 und 4.10.3**).

Die zweite Belastungsversion wurde nicht durch weitere Rechnungen bestätigt. Das ermittelte Ergebnis fußt nur auf einer Rechnung. Es zeigt sich aus den zwei Versionen, dass bei diesem Modell unterschiedliche Belastungen zu unterschiedlichen Ergebnissen (Konstellationen oder Anordnungen) führen können. Die Steifigkeiten des Modells wurden durch die Optimierungen in allen Fällen ganz erheblich erhöht.

		V1 [mm]	V2 [mm]	V3 [mm]	V4 [mm]	Ergebnis-wert [mm]	Einzel-rech-nungen
GradientZurück1	Startwert Endwert	500 30	500 967	500 933,4	500 30	81,04 2,27	215
PlusMinus1	Startwert Endwert	500 50	500 950	500 50	500 950	81,04 2,41	139
GradientZurück1_1	Startwert Endwert	100 38,24	900 962,1	100 41,8	900 959,6	3,19 2,4	120
GradientZurück2	Startwert Endwert	500 961,6	500 30	500 961,6	500 55,8	81,5 1,63	94

Abb. 4.10.3: Ergebniswerte

5. Vergleich und Bewertung der Algorithmen

Bei den behandelten zweidimensionalen Problemen lassen sich per Augenschein zwei markant unterschiedliche entstandene Feldarten beschreiben. Die Unterschiede lassen sich am Verlauf der Sigmalinien festmachen. Diese sind bei drei der vier zweidimensionalen Modellen konvex und die Linien nähern sich im Unendlichen den Achsen des Koordinatensystems an. Bei dem Modell 2 Stäbe entfernt sich allerdings ein Arm der Kurve wieder von der Achse. Die Felder scheinen recht gleichmäßig zu sein, denn die Wege der Algorithmen durch die Felder sind meist ziemlich geradlinig oder nur leicht gebogen. Trotzdem sind Unregelmäßigkeiten vorhanden, und an den Sigmalinien sind lokale Minima zu beobachten (2 Stäbe und Kraftteilung 2 Var).

Bei dem Modell Momentenplatte wurde allerdings ein prinzipiell anderer, ein konkaver Verlauf der Sigmalinie festgestellt. Da die Linie zwei Arme hat, die jeder konkav in einem Punkt sich treffen, erinnert die Linie an eine Karoecke. Dies hatte zur Folge, dass der Algorithmus Vektorprojektion-Sigmakorrektur-TastConst nicht verwendet werden kann, da bei Annäherung an die Karoecke, bei zu großer Schrittweite die Tastschritte zu groß werden konnten, und so schlechte Werte für die Gradientenermittlung ergaben. Funktionell sind hier nur die Algorithmen, die auch die Tastschritte bei Misserfolgen kürzen.

Im Gegensatz zu den zweidimensionalen Feldern, die gut sichtbar gemacht werden können, ist dies bei den größer dimensionalen Feldern nicht mehr möglich. Bei mehrdimensionalen Feldern ist die Klassifizierung der Problemarten schwer durchzuführen, da die Felder, Volumenfelder und Spannungsfelder, schwer zu erfassen sind. Komplexer als die zweidimensionalen sind sie allerdings schon. So können nicht nur an der Sigmalinie lokale Optima festgestellt werden und im Feld leichte Erhöhungen um die problemlos herumgegangen werden kann (**Abbildung 4.1.3**, Reihe 6), sondern es sind Minima und Maxima im Feld so stark ausgeprägt, dass ein falscher Weg zu einem anderen (schlechteren) Optimum eingeschlagen wurde, welches auch mit Vektorprojektion-Sigmakorrektur nicht mehr verlassen wurde (3 Stäbe). Erst die langwierige Untersuchung mit Plus-Minus-Global hat ein besseres Optimum gefunden.

Es wurden zwölf Algorithmen entwickelt.

1. Gruppe

-Halbe Schrittweite
-Goldener Schnitt
-Gradient Zurück
-Gradient Nebenbedingung

2. Gruppe

-Vektorprojektion-Sigmakorrektur
-Vektorprojektion-Sigmakorrektur-TastKonst
-Vektorprojektion-Sigmakorrektur-Mittel
-Penalty-Gradient-Linear
-Penalty-Gradient-Spannungsgewichtet
-Penalty-Vektorprojektion
-Plus-Minus
-Plus-Minus-Global

Die Algorithmen Gradient und Gradient Lang wurden wegen sehr großer Ähnlichkeit in Aufbau und Wirkung zu Gradient-Zurück gezählt. Vektorprojektion-Sigmakorrektur existiert in dreifacher Ausfertigung. Es gibt noch Vektorprojektion-Sigmakorrektur-TastConst, bei dem die Tastschrittweiten auch bei Ausführungsschrittweiten konstant bleiben und Vektorprojektion-Sigmakorrektur-Mittel, bei dem nicht die maximalen Spannungswerte zur Gradientenermittlung herangezogen werden, sondern der Mittelwert aus allen Elementspannungen. Genauso gibt es den Algorithmus Penalty-Gradient dreimal. Zum einen den normalen, dann den spannungsgewichteten und Penalty-Vektorprojektion.

Die Algorithmen sind in zwei Gruppen eingeteilt. Die Unterscheidung fußt auf der Fähigkeit der zweiten Gruppe, einzelne Variablen zu verschlechtern, um mit verbesserten Variablen gleichzeitig das Gesamtergebnis zu verbessern. Diese Eigenschafft bestimmt erheblich die Einsetzbarkeit der Algorithmen. Sie haben die Fähigkeit, sich in alle Richtungen des Feldes zu bewegen. Die der ersten Gruppe sind in ihrer Bewegungsrichtung beschränkt. Dabei gilt allerdings, dass außer Penalty-Gradient-Linear und Plus-Minus-Global die Startwerte im zulässigen Bereich liegen müssen. Zudem sollten die Algorithmen der Vektorprojektion in der Nähe der Sigmalinie gestartet werden, da sie sonst einen unnötigen Zick-Zack-Kurs verfolgen.

Die Algorithmen der zweiten Gruppe bezahlen einen Preis für ihre Fähigkeit. An den Randbedingungen können sie Probleme bekommen. Auch können die einzelnen Variablen mit ihren Einflüssen auf Spannung und Volumen so aufgestellt sein, dass sie die ermittelten Gradienten auf der Stelle treten lassen, fälschlich konvergieren oder falsch laufen.

Diese Einschränkung hat Plus-Minus nicht, dafür kann er aber recht zeitaufwendig in seinem Rechenlauf sein. So behält die erste Gruppe ihre Berechtigung, solange sie Ziel führend ist.

Ein weiterer Unterschied zwischen den Gruppen ist die Gradientenermittlung. Die der zweiten Gruppe ist das klassische Gradientenverfahren. Deren Änderungen der Spannung und des Volumens werden durch die Schrittlänge geteilt und damit auf eins normiert. Bei denen der ersten Gruppe hat man die Möglichkeit durch die Wahl der Schrittlänge Einfluss auf den Gradienten zu nehmen, da nicht mehr durch die Schrittlänge geteilt wird. So bekommt eine Variable durch einen langen Tastschritt die Möglichkeit den Einfluss zu erhöhen, eine höhere Gewichtung zu bekommen und so den Gradienten zu beeinflussen. Mit dem Modell 2 Stäbe wurden beide Gradientermittlungsarten verglichen. Dabei hat sich gezeigt, dass die ermittelten Gradienten auch sehr ähnlich sein können. Das heißt, trotz ungleicher Schrittlängen, wird der Gradient unter Umständen nur sehr wenig verformt.

Auf der anderen Seite hat man die Möglichkeit, durch eine sehr lang gewählte Schrittlänge (Modell Kraftteilung 4 Var) einen Gradienten zu ermitteln, mit dem der gesamte Definitionsbereich der Variable 1 zu durchfahren, bis das globale Optimum erreicht wurde, was mit den anderen Gradientenverfahren nicht möglich gewesen wäre, zumindest aber nicht in angemessener Zeit, da der Einfluss der Stabradien so dominant ist, dass immer diese zu sehr verdünnt werden und nicht die Variable 1 verändert würde.

Was bei diesem Beispiel allerdings auch noch greift, ist das Problem einer entkoppelten Variablen. Dies ist die Variable 3, also die Dicke des oberen einzelnen Stabes. Entkoppelt bedeutet, dass seine Belastung nichts mit den anderen Variablen und deren momentaner Größe zu tun hat. Natürlich hat dann diese Variable eigentlich auch nichts mit der Optimierung zu tun und müsste aus der Optimierung herausgelöst werden, allerdings kann es sein, dass es auch nur teilentkoppelte Variablen gibt, die nur von einigen Variablen unabhängig sind und trotzdem zum Problem gehören. Deshalb ist es interessant, dieses Modell in Gänze zu lösen. Wie schon erwähnt, hat man die Möglichkeit dies mit Gradient-Zurück durchzuführen, weiterhin ist Plus-Minus dazu in der Lage. Allerdings muss, da ein lokales Minimum entdeckt wurde, von da aus mit Plus-Minus-Global gearbeitet werden, um mit Schrittweitenvergrößerung das Minimum zu verlassen, oder man muss von Anfang an mit genügend großen Schrittweiten arbeiten. Weiterhin ist Vektorprojektion-Sigmakorrektur-Mittel in der Lage solch eine entkoppelte Variable zu verarbeiten, allerdings, wie schon erwähnt, bei diesem Modell wegen seiner Variableneinflussgrößen nur sehr langsam.

An dem Modell Kraftteilung4Var lässt sich ein weiteres Problem festmachen. In diesem Beispiel gibt es eine abhängige Variable. Dies ist die Variable 1, also die

Platzierung des Bleches und dadurch bedingt, die Längen der Stäbe. Diese Variable hat allerdings keinen eigenen Körper, sondern die Volumenänderung entsteht durch die Änderung der Volumen der Stäbe, die allerdings eher den Variablen der Radien der Stäbe zugeordnet werden. Dies muss kein Problem sein, aber bei der Ermittlung der Gradienten kann es zu einem Problem werden. Die ermittelte Änderung bei Änderung der Variablen 1 kann durch die folgende Änderung der Radien der Stäbe sich als falsch herausstellen. Dies ist zwar ein grundsätzliches Problem bei dieser Art der Sensitivitätsanalyse, aber in diesem Fall ist durch das fehlende eigene Volumen die Gefahr brisanter. Die Lösung des Problems ist in so einem Fall der Algorithmus Plus-Minus, da er gleichzeitig Variablen ändert und daraus seine Schlüsse zieht.

Die mehrdimensionalen Probleme lassen sich ein zwei Gruppen teilen. Zum eine diejenigen, die aus einem Körper bestehen, das heißt, jede Variable hat einen direkten Einfluss auf diesen Körper. Dies sind die beiden Pleuel-Versionen und das Modell RePlatte3Var. Für diese ist Vektorprojektion-Sigmakorrektur wie geschaffen. Anstandslos läuft er lange Wege zügig an der Sigmafläche entlang und bleibt an keinem lokalen Optimum hängen. Plus-Minus funktioniert auch, ist aber langsamer und am Ende nicht ganz so genau. Auch Gradient-Zurück kommt mit diesen Problemen zurecht, bleibt aber, je nach der Startwertwahl irgendwo vor dem Optimum an der Sigmafläche hängen.

Die schwierigere Problemform sind die getrennten Volumina. Hierbei sind einzelne Variablen für einzelne Volumen zuständig, z.B. die Modelle 3Stäbe, Kraftteilung4Var und Momentenplatten. Hier funktionieren zwar auch die Gradientenverfahren, zum Teil auch ziemlich gut, trotzdem ist die Gefahr von lokalen Optima, die anscheinend sehr weit reichen, d.h. schon sehr früh geht der Algorithmus auf eine Variablenkonstellation zu, aus der er am Ende auch nicht mehr herauskommt. Erst eine langwierige Untersuchung mit Plus-Minus-Global bringt ein besseres Ergebnis, oder, wie bei Momentenplatten, bringt Plus-Minus das beste Ergebnis.

Eine interessante und sehr zügige Alternative bietet die Penalty-Vektorprojektion. Diese läuft bei einigen Modellen sehr schnell an der Sigmafläche entlang, weil im Gegensatz zu Vektorprojektion-Sigmakorrektur die zeitaufwendigen Korrekturschritte wegfallen. Leider muss man sagen, eben weil die Korrektur wegfiel und wegen der konvexen Sigmafläche, befand sich der Algorithmus meist im unzulässigen Bereich, den er auch nicht mehr verlassen konnte. Jedoch beginnt in der Nähe des Optimums ein „auf der Stelle Treten“. Dann kann mit dem Algorithmus Penalty-Gradient-Linear der unzulässige Bereich verlassen werden und das Optimum erreicht werden.
Weiter muss man sagen, dass bei mehrdimensionalen Problemen die Penalty-Verfahren Schwächen zeigten oder zumindest sehr langsam waren. Um die Bewegung entlang der Sigmafläche zu beschleunigen, besteht noch Entwicklungsbedarf.

Die zweidimensionalen Modelle waren für sämtliche Algorithmen gut lösbar. Entsprechen ihrer Spezifikationen waren sie zwar prädestiniert oder gebremst, aber alle konnten gute Lösungen erbringen.

Die Algorithmen der ersten Gruppe und die Algorithmen Plus-Minus und Plus-Minus-Global sind in der Lage, auch andere Probleme als das der restringierten Gewichtsminimierung zu lösen. Die Einsetzbarkeit wurde zufrieden stellend an dem Modell Stabtragwerk überprüft.

Einige mögliche Strategien zur Durchführung der Optimierungen haben sich bewährt. Eine Möglichkeit ist mit Plus-Minus-Global und recht großen Schrittweiten zu beginnen, um so einen Überblick zu gewinnen und mit einigen interessanten Werten mit anderen Algorithmen fortzufahren. Wenn dass zu langwierig erscheint, kann auch versucht werde, mit einem Gradientenverfahren der 1.Gruppe, oder auch mit Penalty-Gradient-Linear einen Weg zur Sigmafläche zu gewinnen und von dort mit Gradientenverfahren der 2.Gruppe oder mit Plus-Minus weiterzumachen. Sollte man ein lokales Minimum vermuten, kann mit Plus-Minus-Global mit kleinen Schrittweiten an der Stelle geprüft werden und bei Erfolg auch noch weiter mit Gradientenverfahren gerechnet werden.

Weiterhin wurde die Methode der zwei Ergebnissets erfolgreich getestet. Um die Elemente an festen Einspannungen oder rechten Winkeln und die dort theoretisch unendlich hohen Spannungen aus der Bewertung herauszunehmen, kann in der Optimierung mit zwei Ergebnissets gearbeitet werden. Das eine Set beinhaltet alle Elemente zur Volumenermittlung. Zur Spannungsermittlung müssen die Elemente an der Einspannung herauspartitioniert werden. Die Elemente ohne die der Partitionierung bilden das zweite Set.

Untersucht wurde auch das Problem von abweichenden Spannungen durch unterschiedliche Netze durch die automatische Neuvernetzung. Dieses Problem scheint jedoch nicht oft und meistens nicht schwerwiegend aufzutreten. Festgestellt wurde es bei dem Modell Kraftteilung2Var. Dort wurden Stäbe kubisch vernetzt und bei einer Rechnung wurde die Gradientenermittlung zu ungenau, so dass sich die Rechnung schon recht nahe am Optimum festgelaufen hat. Dies Problem konnte jedoch durch Änderung der Schrittweite einer der beiden Variablen vermieden werden (geändertes Schrittweitenverhältnis). Weiterhin wurde festgestellt, dass bei dem Modell RePlatte3Var sehr hohe Spannungswerte ermittelt wurden, wenn die Vernetzung der Platte nur eine Elementreihe (Schalenelemente) aufwies. Diese hohen Werte verhinderten eine sinnvolle Optimierungsrechnung. Erst als darauf geachtet wurde, dass mindestens zwei Reihen erstellt wurden, konnte das Optimum gefunden werden.

6. Zusammenfassung

In vielen technischen Entwicklungsprozessen wird der Punkt erreicht, an dem ein physikalisch-mathematisches Modell des zu lösenden Problems besteht, das aber den gestellten Anforderungen noch nicht genügt. Prinzipiell soll das Modell nicht mehr geändert werden, aber einige der Größen, die das Modell beschreiben, müssen geändert werden, um die Zielvorgaben zu erfüllen. Meistens beeinflussen sich diese Größen gegenseitig. Die Suche nach der besten Lösung kann dann sehr aufwendig und schwierig sein. Für diesen Fall wurde in dieser Arbeit eine Optimierungsumgebung entwickelt.

Die zu lösenden Probleme gehören zum Bereich der Strukturmechanik. Dort werden sehr oft computerunterstützte Konstruktionsmittel zu Hilfe genommen. Dies sind meistens kommerzielle Programme der Methode der Finiten Elemente, da sich mit ihnen eine große Zahl unterschiedlicher strukturmechanischer Probleme näherungsweise lösen lassen. Die entwickelte Optimierung basiert auf solch einem FEM-Programm und den Modellen, die mit ihm erstellt wurden. Damit ist es möglich, die vielseitigen Anwendungsmöglichkeiten eines FEM-Programms direkt für die Optimierung zu nutzen. Sämtliche Änderungen, die in einem FEM-Modell durchgeführt werden können, können auch in die Optimierung einfließen. Dies macht die Variationsmöglichkeiten für die Optimierung sehr groß.

Das genutzte FEM-Programm ist Abaqus mit seiner graphischen Oberfläche CAE. Mit der Programmiersprache PYTHON besteht die Möglichkeit, die Oberfläche CAE zu verändern und diesen Veränderungen neue Funktionalitäten zuzuweisen. So wurde die Oberfläche um einen Bereich „Optimierung“ erweitert, der aus zusätzlichen Fenstern, die geöffnet werden können, besteht. Mit diesen Fenstern wird der Anwender durch die Optimierung geführt. Das bedeutet, dass die zu variierenden Größen, die Parameter, bezeichnet werden müssen und zur Durchführung des Optimierungslaufes notwendige Größen, wie Startschrittweiten und Abbruchgrenzen, eingegeben werden. Aus diesen Informationen setzt das Programm ein Script, einen neuen Quelltext, zusammen. Dieser kann gestartet werden und damit auch der Ablauf der Optimierung. Das bedeutet, dass in der Folge einzelne FEM-Rechnungen ausgeführt werden und aus der jeweiligen Ergebnisermittlung und Auswertung Schlüsse gezogen werden, aus denen neue Werte für die einzelnen Variablen ermittelt werden. Dies wird solange wiederholt, bis das Abbruchkriterium erreicht wird. Die Größen des Modells, die geändert werden sollen, heißen Parameter

wodurch die Optimierungsumgebung zu einer Parameteroptimierung wird. Die Parameter verändern einzelne Durchmesser, Längen und die Form des Modells. Das Programm ist variablenunabhängig, d. h. jede Zahl an Variablen kann berechnet werden. Ausgelegt ist es jedoch für eine Variablenzahl von bis zu zehn Variablen.

Kernstück jeder Optimierung sind die ihr zugrunde liegenden Algorithmen. Für die Optimierung dieser Arbeit wurden zwölf Algorithmen entwickelt, die zu den Suchverfahren, die einen Tastzyklus durchlaufen und davon den besten Wert auswählen, und zu den Gradientenverfahren 1.Ordnung gehören. Mit diesen Algorithmen können unrestringierte Probleme bearbeitet werden und solche mit einer Restriktion. Die Algorithmen sind für die Strukturoptimierung von Bauteilen spezifiziert. Dazu wurden mehrere Bauteilmodelle erstellt, anhand derer die Algorithmen entwickelt und überprüft wurden. Zehn dieser Modelle und deren beispielhafte Optimierungen sind in dieser Arbeit aufgeführt. Die einzelnen Algorithmen haben im Optimierungsablauf unterschiedliche Aufgaben und sind für manche Probleme besser geeignet als andere. Drei Aufgaben lassen sich unterscheiden. Überprüfung des gesamten oder eines großen Teils des Definitionsbereiches der Variablenwerte. Dabei sollen aussichtsreiche Variablenkonstellationen ermittelt werden, die als Startwerte für weitere Optimierungsrechnungen dienen können. Die zweite Aufgabe ist der Weg durch das Feld der Zielfunktion bis zur Restriktionsfläche. Danach folgt der Weg entlang der Restriktionsfläche bis zu dem Optimum. Sollten Zweifel bestehe, dass es sich dabei um das globale Optimum handelt, lässt sich diese Aufgabe um die Suche nach dem globalen Optimum erweitern. Es gibt Problemstellungen (entkoppelte Variable, abhängige Variable, Neuvernetzung), bei denen die über die Gradienten ermittelten Ausführungsschritte nicht genau genug sind und deshalb die Gradientenverfahren Probleme bekommen können, zum Ziel zu führen. Das Suchverfahren Plus-Minus hat diese Probleme nicht. Dies hat allerdings den Preis der größeren Zahl an benötigten Rechnungen und damit des größeren Zeitaufwands. Die Gradientenverfahren dagegen sind in der Lage, schnell zum Ziel zu kommen, können jedoch bei manchen Problemstellungen behindert werden den Gradienten zu ermitteln.

Um diese Lücke zu schließen, könnten in Zukunft die Suchverfahren weiterentwickelt werden. Sie können in ihrer Grundstruktur bestehen bleiben, sollten aber die Möglichkeit bekommen, aus vorherigen Ergebnissen Schlüsse zu ziehen, um zuerst die Erfolg versprechenden Suchrichtungen zu untersuchen und dadurch den Rechenaufwand zu verkleinern. Weiterhin könnte bei sämtlichen Algorithmen ein Datenhaltungs- und Bewertungssystem implementiert werden, das weiß, ob eine Variablenkonstellation oder eine ihr ähnliche Konstellation im Optimierungslauf schon gerechnet wurde um dann die ermittelten Werte zu verwenden und so Rechnungen einzusparen.

Anhang A: Die GUI-Erweiterung

Die Optimierung dieser Arbeit ist auf der Basis der CAE-Erweiterung GUI entstanden. Die prinzipielle Funktionsweise von GUI und ihr modularer Aufbau sollen in diesem Anhang verdeutlicht werden.

A.1 Abaqus/CAE

Der *Python Interpreter* übergibt den Python Code an den Abaqus/CAE Kern, der daraus ein *Input-File* erstellt. Ein Input-File ist die ursprüngliche Form der Erstellung eines FEM Modells durch die so genannte *Makroprogrammierung*. Diese Form ist für den Rechencode von Abaqus lesbar und ist die Grundlage der Durchführung der FEM-Rechnung (Abaqus).

Die graphische Oberfläche von Abaqus/CAE ist mit dem *Abaqus GUI Toolkit* erstellt worden bzw es besteht aus dessen Klassen. Dieses Toolkit ist eine Erweiterung des Fox GUI Toolkit, welches eine plattformunabhängige objektorientierte GUI-Klassenbibliothek ist und die wichtigsten GUI-Bausteine enthält. Aus diesem Toolkit und seinen Klassen setzt sich die Oberfläche zusammen.

Die GUI-Datenbanken sind als Klassen strukturiert, d.h. sie sind objektorientiert programmiert. Dadurch ist es möglich, dass man auf die Programmierung von Abaqus/CAE in einem gewissen Rahmen zugreifen und sie verändern kann. Genauer gesagt, startet man seine eigene *Applikation*, also sein eigenes Programm, das aber die vererbten Klassen des originalen CAE übernimmt. Was dabei herauskommt, ist die gewohnte CAE-Oberfläche, die mit eigenen Modifikationen, Fenstern und auch Funktionalitäten erweitert wurde.

Die Programmierung von CAE ist nicht nur objektorientiert, sondern auch modular strukturiert. Für die äußerliche Änderung von CAE, müssen Klassen übernommen werden. Dies geschieht in Modulen. Auch die neuen Funktionalitäten werden als Module dem Quellcode zugefügt. Diese neuen Module, die Applikationsmodule, können den Ordnern der GUI-Bibliotheksdateien zugeordnet werden, oder sie werden im Arbeitsverzeichnis gespeichert.

Der Aufruf dieser Applikation geschieht in der Abaqus Kommandobox. Nun wird der Aufruf abaqus cae erweitert und sieht in der Gesamtheit so aus:

Abaqus cae – custom optiApp

wobei optiApp der variable Name des aufzurufenden Moduls ist.

A.2 Die Klassen der Gui-Erweiterung

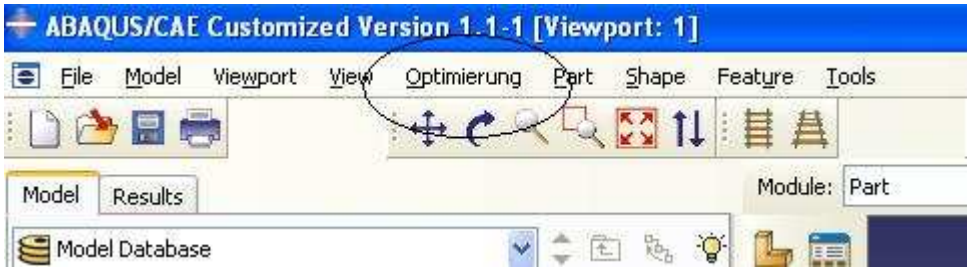


Abb. A.2.1: Optimierung im Hauptfenster

Die Erweiterung von Abaqus/CAE äußert sich im Erscheinen eines neuen Reiters in der Menüleiste des Hauptfensters (**Abbildung A.2.1**). Dieser Reiter hat den Namen Optimierung. Dabei handelt es sichi um ein Toolset, das durch den erweiterten Aufruf von Abaqus/CAE gestartet wird. Das Python-Modul optiApp.py ist das übergeordnete Skript, das die ihm folgenden Module aufruft, die das Optimierungswerkzeug aufbauen (**Abbildung A.2.2**).

```
app = AFXApp('ABAQUS/CAE', 'ABAQUS, Inc.')
app.init(sys.argv)


OptiMainWindow(app)


app.create()
app.run()
```

Abb. A.2.2: optiApp.py

Mit app wird eine Instanz der Klasse AFXApp erzeugt und es wird eine Verbindung zum Kern aufgebaut. Die dabei übergebenen Argumente sind die Registrierungsschlüssel für die Applikation und den Hersteller. OptiMainWindow ist ein Instanzobjekt und es ist für die Komponenten des Hauptfensters zuständig. Ihm wird app als Argument übergeben. Das Fenster für die Applikation wird mit der

Methode *create* erzeugt, welche von der Basisklasse FXApp geerbt ist. Die Applikation ist eine endlos-Schleife und wird mit der run-Methode (FXApp) gestartet.

```
class OptiMainWindow(AFXMainWindow):

   def __init__(self, app, windowTitle=''):

      AFXMainWindow.__init__(self, app, windowTitle)

        ...
      self.registerToolset(OptiToolsetGui(), GUI_IN_MENUBAR|GUI_IN_TOOLBAR)
      self.registerToolset(TreeToolsetGui(), GUI_IN_MENUBAR|GUI_IN_TOOLBAR)

      self.registerModule('Part',          'Part')
      self.registerModule('Property',      'Property')
      self.registerModule('Assembly',      'Assembly')
        ...
```

Abb. A.2.3: Class OptiMainWindow

Die *AFXMainWindo-Klasse* ist für die Gestaltung der Komponenten des Hauptfensters konzipiert. Von ihr erbt die *OptiMainWindow-Klasse* (**Abbildung A.2.3**) ihre Eigenschaften. Objekte, die im Hauptfenster ständig zur Verfügung stehen, werden mit der *registerToolset-Funktion* gebildet. Objekte, die erst während der Anwendung ausgewählt werden, wie die Module Part, Property usw. werden mit der *registerModule-Funktion* erstellt.

```
class OptiToolsetGui(AFXToolsetGui):

   def __init__(self):

      AFXToolsetGui.__init__(self, 'Opti Module')

      menu = AFXMenuPane(self)
      AFXMenuTitle(self, '&Optimierung', None, menu)
      AFXMenuCommand(self, menu, '&StartLern...', None,
         OptiStartAnalyzeForm(self), AFXMode.ID_ACTIVATE)
      AFXMenuCommand(self, menu, '&EndeLern...', None,
         OptiEndAnalyzeForm(self), AFXMode.ID_ACTIVATE)

        subMenu1 = AFXMenuPane(self)
        AFXMenuCascade(self, menu, '&HalbeSchrittweite', None, subMenu1)
        AFXMenuCommand(self, subMenu1,'&Modulerstellung',None,
            OptiModulerstellungHalbeSchrittweiteAnalyzeForm(self),
            AFXMode.ID_ACTIVATE)
        AFXMenuCommand(self, subMenu1, '&Berechnung', None,
            OptiBerechnungHalbeSchrittweiteAnalyzeForm(self),
            AFXMode.ID_ACTIVATE)
        ....
   def getKernelInitializationCommand(self):

      return 'import optiModul'
```

Abb. A.2.4: class OptiToolsetGui

Das Abaqus Gui Toolkit basiert auf den Klassen des Fox Toolkit (FX) und den für Abaqus erweiterten Klassen (AFX). Dazu gehört die Klasse *AFXToolsetGui*. Von dieser erbt die Klasse *OptiToolsetGui* (**Abbildung A.2.4**).

Allgemein werden in Klassen Methoden definiert und Methoden der vererbenden Klasse überschrieben. Mit der _init_Funktion wird eine neue Instanz einer Klasse erzeugt. Hier wird die Instanz *OptiToolsetGui* initiiert. Dann wird mit *AFXToolsetGui._init_* eine neue Instanz der gleichnamigen Klasse erzeugt. Nach ihr wird eine Instanz der Klasse *AFXMenuPane* erzeugt. Mit ihr werden die alternativen Unterpunkte eines Menüreiters beschrieben. Den Menüreiter beschreibt die AFXMenüTitle-Instanz. Die Instanzen der Klasse AFXMenuCommand beinhalten die Beschriftung der Menüreiter und die Klassen, die sie für die nachfolgenden Dialoge aufrufen (OptiBerechnungHalbeSchrittweiteAnalyzeForm). Mit der Funktion

getKernelInitializationCommand wird das Modul *OptiModul.py* im Kern initialisiert. Dessen Kommandos sind maßgeblich für das Erstellen und das Ausführen der Optimierung verantwortlich.

A.3 Zugriff auf die Ergebnisdatei

Die Ergebnisse einer Abaqus-Rechnung werden in einer Ausgabedatei (*Dateiname*.odb) gespeichert. Sie beinhaltet sowohl Modell- als auch Ergebnisdaten. Diese Daten können mit Abaqus/CAE grafisch dargestellt werden.

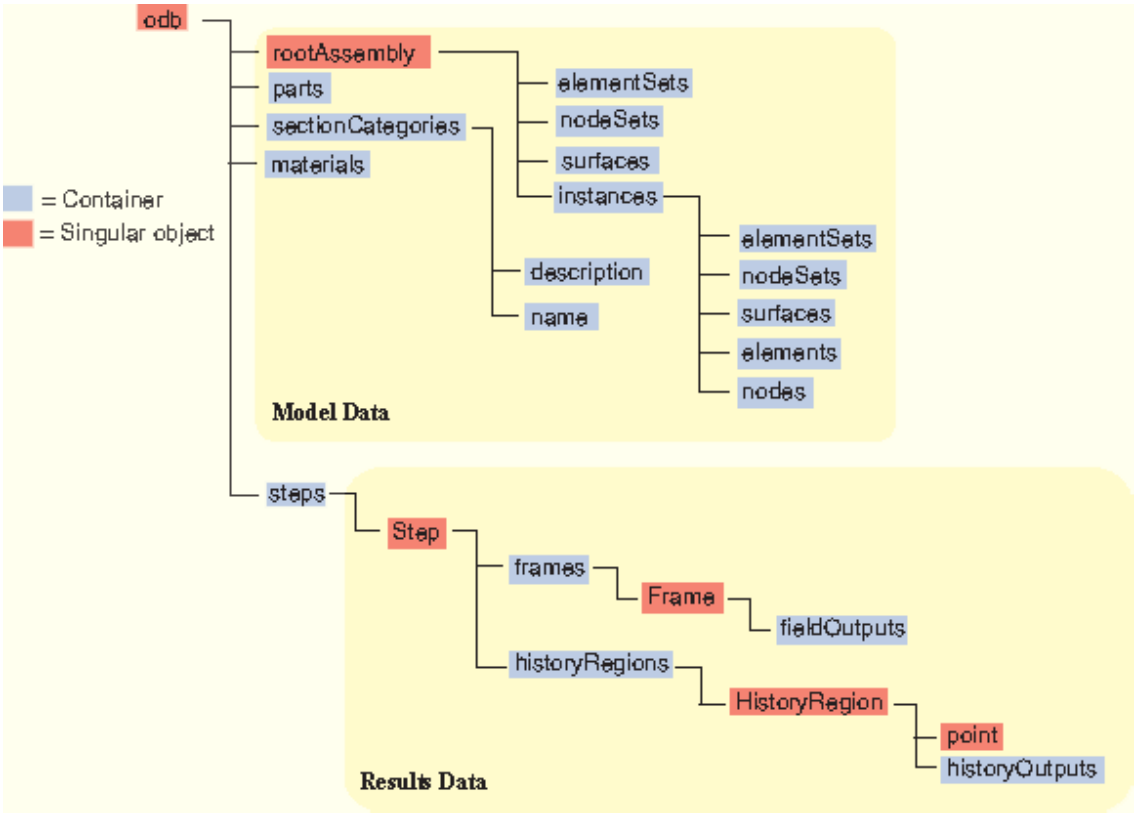


Abb. A.3.1: Struktur des Objektmodells der Ausgabedatei [Aba10]

Für den Ablauf der Optimierung ist es aber notwendig, einzelne Ergebniswerte als float-Vaiablenwert zu erhalten und automatisch dem Programm zur Verwertung zur Verfügung zu stellen. Mit PYTHON und den passenden Befehlen (Skripting) hat man die Möglichkeit dazu. Um auf ein Objekt der Ausgabedatei zuzugreifen, wird die Struktur des Objektmodells verwendet (**Abbildung A.3.1**). Die Abbildung zeigt den Aufbau der Ausgabedatei. Sie ist in die zwei Bereiche *Modell Data,* in denen die definierten Sets aufgelistet sind,und *Results Data,* in denen jegliche Ergebniswerte gespeichert sind, aufgeteilt. Zu allen Sets lassen sich damit alle Ergebniswerte auslesen. Dies ist im folgenden Abschnitt beispielhaft ausgeführt.

In **Abbildung A.3.2** wird auf die Verschiebung eines definierten Sets zugegriffen. Zuerst wird die entsprechende Ausgabedatei identifiziert und geöffnet. Center identifiziert den gewählten Set in der Ausgabedatei und bestimmt ihn als *Zielset,* dessen Werte herausgeschrieben werden. Zum Schluss wird bestimmt welche Werte (Verschiebung) in welchem Schritt herausgeschrieben werden. Dieser Wert wird der Variablen *Ergebniswert* zugeordnet und damit der Optimierung zur Bearbeitung übergeben.

```
odb = openOdb(path='./%s.odb' % JobName)
center = odb.rootAssembly.instances[InstanceName].nodeSets[SetName]
Ergebniswert = odb.steps[StepName].frames[-1].lastFrame.fieldOutputs['U'].
        getSubset(region=center).values[0].magnitude
```

Abb. A.3.2: Befehl zur Extraktion von Ausgabedateiwerten

Literatur

[Aba 10] Abaqus Online Documentation 2010

[And 03] Andrä, H., Spezifische Strukturoptimierungsverfahren für Gießereien, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[And 05] Andrä, H., OPTCAST- Entwicklung adäquater Strukturoptimierungsverfahren für Gießereien, Berichte des Fraunhofer ITWM, Nr. 80, 2005

[Bai 94] Baier, H., Optimierung in der Strukturmechanik, Vieweg 1994

[Ben 95] Bendsöe, M., Optimization of Structural Topology, Shape and Material, Springer, 1995

[Ble 03] Bletzinger, K., Mathematische Agorithmen in der nichtlinearen Programmierung, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Bog 03] Boggasch, M., Anwendung neuer Methoden bei der Topologie- und Formoptimierung, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Bom 93] Bomze, M., Optimierung-Theorie und Algorithmen, Wissenschaftsverlag 1993

[Bös 06] Bös, J.; Numerical optimisation of the thickness distribution; Struct Multidisc Optim (2006) 32: 12-30

[Cha 08] Chang, Y., Shape Optimization of Mufflers, Int. J. Meth. Engng 2008, 74: 1592-1620

[Che 07] Chen, J., Shape optimization with topological changes, Int. J. Numer. Meth. Engng 2007, 71: 313-346

[Chr 09] Christensen, P., An Introduction to Structural Optimization, Springer, 2009

[Coe 08] Coelho, P., A hierarchical model for concurrent material and topology optimisation, Struct Multidisc Optim, 2008, 35: 107-115

[Ege 07] Egerland, M., Optimization of a fan shroud, ANSYS Conference & 25th CADFEM User's Meeting, November 2007, Dresden

[Els 78] Elster, K., Nichtlineare Optimierung, Verlag Harri Deutsch Frankfurt 1978

[Esc 06] Eschenauer, H.; Structural behaviour and optimal layout of light constructions; Arch Appl Mech (2006) 75: 441-457

[Fis 03] Fischer, W., Bauteiloptimierung mit FEM und CAD, Gießerei 90, 6/2003

[Fis 04] Fischer, W., Bauteiloptimierung mit Hilfe der FEM, Congress Intelligente Leichtbausysteme, Sept 2004

[Grü 03] Grün,F.;Eichsleder; W.; Form- und Topologieoptimierung unter Berücksichtigung der Betriebsfestigkeit;XXII. Verformungskundliches Kolloquium-Tagungsband, Planneralm 2003

[Har 06] Harzheim, L., A review optimization of cast parts, Struct Multidisc Optim, 2006, 31: 388-399

[Har 08] Harzheim, L., Strukturoptimierung, Grundlagen und Anwendungen, Harri Deutsch Verlag, 2008

[Häu 03] Häußler, P., Entwicklung und Konstruktion von innovativen Leichtbauprodukten, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Hil 88] Hillier, F., Operations Research, Oldenbourg Verlag München 1988

[Hop 04] Hoppe, A., Multidisziplinäre optimierung parametrischer Fahrzeug-komponenten, VDI Berichte Nr. 1833, 2004

[Hör 04] Hörnlein, H.; Sensitivitätsanalyse und Rechenzeiten in der Strukturoptimierung, Technisch-wissenschaftliches Seminar-Optimierungsprozesse bei der Entwicklung von Leichtbaustrukturen UniBw München, 9. Dezember 2004

[Kel 09] Kellermeyer, M., Fitting of Parameters with optiSlang, ANSYS Conference & 27th CADFEM User's Meeting, November 2009, Leipzig

[Köt 03] Köttgen, V., Optimierung im Hinblick auf Betriebsfestigkeitseigenschaften, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Lau 03] Lauber, B., Berücksichtigung von fertigungsbedingten Restriktionen in der Gestaltoptimierung, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Lau 08] Lauber, B., Using Tosca Structure in the Ansys Environment, ANSYS Conference & 26th CADFEM User's Meeting, Oktober 2008, Darmstadt

[Laum 02] Laumanns, M., Evolutionäre Algorithmen in der Fahrzeugentwicklung, 11. Aachener Kolloquium Fahrzeug- und Motorentechnik 2002

[Lit 92] Littger, K., Optimierung, Springer Heidelberg 1992

[Lut 09] Lutz, A.,ACP & optiSlang, ANSYS Conference & 27th CADFEM User's Meeting, November 2009, Leipzig

[Mar 06] Marti, K., Stochastische Strukturoptimierung von Stab- und Balkentragwerken, Springer, 2006, Heidelberg

[Mer 95] Merziger, G., Repetitorium der höheren Mathematik, Binomi Verlag Springe, 1995

[Mes 00] Meske, R., Topologie- und Gestaltoptimierung mit CAOSS und ABAQUS, ABAQUS Anwendertreffen 2000, Winterthur

[Nit 05] Nitsopulos, I., Multidisziplinäre Optimierung mit ABAQUS und OPTIMUS, ABAQUS User's Conference, AT_Graz, September 2005

[Pad 04] Padmanaban, R., Mesh Morphing-based Shape Optimization of a Clutch Lever, 2004 Abaqus User's Conference, May 25-27, Boston

[Pap 91] Papageorgiou, M., Optimierung, Oldenbourg Verlag München, 1991

[Ped 08] Pedersen, P., Suggested benchmarks for shape optimization, Struct Multidisc Optim, 2008, 35: 273-283

[Pha 07] Pham, T., Toleranzanalyse und Optimierung eines Finite-Elemente-Strukturmodells mit OptiY, 19.Deutschsprachige Abaqus benutzerkonferenz, Sept 2007, Baden-Baden

[Puc 03] Puchner, K., Strukturoptimierung auf Basis von Lebensdauer-ergebnissen, Symposium Simulation in der Produkt- und Prozessentwicklung, Nov 2003, Bremen

[Qua 05] Quadbeck, M.; Struckturoptimierung für Chrashlastfälle anhand parametrischer Flächen- und assoziativer Berechnungsmodelle;LS-DYNA Anwenderforum, Bamberg 2005

[Roo 04] Roos, D.; Optimierung und Sensitivitätsanalyse zur Parameteridentifikation in der numerischen Simulation der Gefügekinetik im Schweißprozess, 22nd CAD-FEM User's Meeting 2004

[Sau 09] Sauter, J., Lenz, C., Lauber, B., From Setup to automatic Validation, ANSYS Conference & 27th CADFEM User's Meeting, November 2009, Leipzig

[Scä 08] Schäfer, C.; Shape Optimisation by Design of Experiments and Finite Elements; Struct Multidisc Optim [2008] 36:477-491

[Sce 10] Scherer, M., A fictitious energy approach for shape optimization, Int. J. Meth. Engng 2010, 82: 269-302

[Scu 05] Schumacher, A. Optimierung mechanischer Strukturen, Springer 2005

[Scwa 01] Schwarz, S., Sensitivitätsanalyse und Optimierung bei nichtlinearem Strukturverhalten, Diss., 2001, Stuttgart

[Scwe 75] Schwefel, H., Evolutionsstrategie und numerische Optimierung, Berlin 1975

[Scwe 77] Schwefel, H., Numerische Optimierung von Computer-Modellen, Birkhäuser Verlag, Stuttgart 1977

[Sem 06] Semler, F., Applications of Design Optimization Using modeFrontier, 24th CADFEM User's Meeting, October 2006, Stuttgart

[Shi 09] Shimoda, M., A practical solution to the shape optimization problem of solid structures, WIT Transactions on the Built Environment, Vol 106, 2009 WIT Press

[Sid 04] Sidhu, R., Shape Optimization of Structures Using Abaqus, 2004 Abaqus User's Conference, May 25-27, Boston

[Smi 97] Smith, O., Topology optimisation of trusses with local stability constraints; Structural Optimisation 13, 155-166, Springer 1997

[Spe 93] Spellucci, P., Numerische Verfahren der nichtlinearen Optimierun, Birkäuser Verlag Basel 1993

[Wil 06] Will, J., Interfacing optiSlang and modeFrontier with Ansys and LSTC Software Products, 24th CADFEM User's Meeting, October 2006, Stuttgart

[Wil 08] Will, J., Robust Design Optimization with optiSlang, ANSYS Conference & 26th CADFEM User's Meeting, Oktober 2008, Darmstadt

[Zha 08] Zhang, J., Structural dynamic shape optimization, Struct Multidisc Optim, 2008, 36: 307-317

[Zhu 08] Zhu, P., Optimum design of an automotive inner door panel, Proc. IMechE Vol. 222 Part D: J. Automobile Engineering 2008

Lebenslauf

Persönliche Daten

Markus Riem
Geburtsdatum: 13. November 1969
Geburtsort: Stuttgart
Familienstand: verheiratet, drei Kinder

Schulausbildung

1976_1980	Grundschule Endersbach
1980 – 1989	Konrad-Heeresbach-Gymnasium in Mettmann Abschluss: Abitur

Wehrdienst

06 / 1989 – 06 / 1991	Ausbildung zum Offizier der Reserve der Bundeswehr

Studium

06 / 1991 – 08 / 1991	Grundpraktikum bei Mercedes-Benz im Werk Düsseldorf
10 / 1991 – 12 / 2002	Studium des Maschinenbaus an der Universität Hannover Diplomarbeit und Fachpraktikum bei der Salzgitter AG

Beruf

08 / 2003 – 09 / 2003	Wissenschaftliche Hilfskraft am Institut für Mechanik der Universität Hannover
10 / 2003 – 04 / 2004	Berechnungsingenieur bei der IVM – Automotive Wolfsburg GmbH in der Abteilung Systementwicklung Integration / Technische Berechnung
06 / 2004 – 09 / 2004	Berechnungsingenieur bei der OSB AG Stuttgart
10 / 2004 – 01 / 2011	Wissenschaftlicher Angestellter am Institut für Maschinenbau der Fachhochschule Flensburg

Freienwill, im Januar 2011