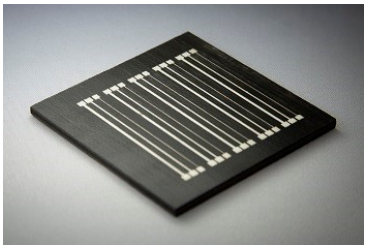
Aufgabenstellung

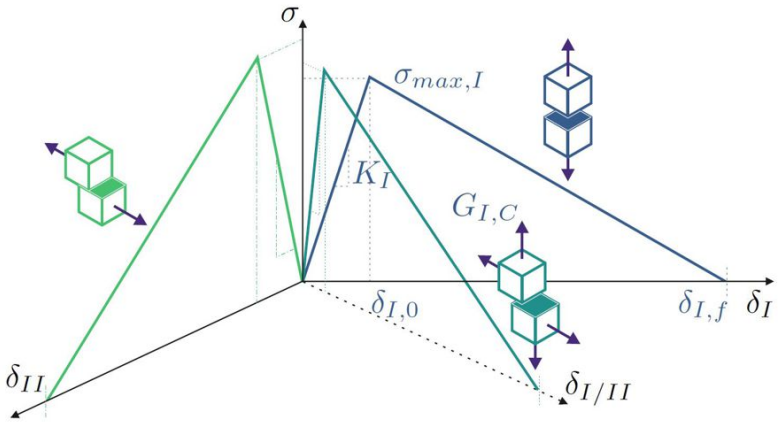
Implementierung eines Optimierungsalgorithmus zur inversen Parameteridentifikation von Kohäsivzonenmodellen

Kategorie: Numerisch

Beschreibung:
In der Luftfahrt können gedruckte Leiterbahnen statt Kabeln verwendet werden, um Gewicht zu sparen und gleichzeitig das Flugzeug mit einem Sensornetzwerk zu versehen. Dieses Sensornetzwerk soll später analog zum menschlichen Nervensystem in quasi-Echtzeit Daten über den Gesundheitszustand der Struktur liefern. Da die aufgebrachte Sensorik im Betrieb Schädigungen unterliegen kann, soll das Materialversagen der Grenzfläche zwischen Sensorik und Flächentragwerk numerisch abgebildet werden.



Im Rahmen dieser Arbeit soll ein Optimierungsalgorithmus zur inversen Parameteridentifikation von Kohäsivzonenmodellen entwickelt und implementiert werden. Dazu soll zunächst eine Literaturrecherche durchgeführt werden. Die Konzeption und Implementierung geschieht mit der FE Software FEAP. Anschließend soll der Algorithmus validiert und dokumentiert werden.



Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

Nomenklatur

Abkürzungen

CFK	Carbonfaserverstärkter Kunststoff
COA	Crack Opening Angle
COD	Crack Opening Displacement
CTOA	Crack Tip Opening Angle
CTOD	Crack Tip Opening Displacement
DCB	Double Cantilever Beam
EPBM	Elastisch-plastische Bruchmechanik
ESZ	Ebener Spannungszustand
EVZ	Ebener Verzerrungszustand
FEAP	Finite Element Analysis Program
FEM	Finite Elemente Methode
KZM	Kohäsivzonenmodell
LEBM	Linear-elastische Bruchmechanik
SIF	Spannungsintensitätsfaktor
TSL	Traction Separation Law

Griechische Formelzeichen		**Einheit**
δ	Rissöffnung	mm
δ_0	Separation	mm
δ_c	kritische Rissöffnung	mm
δ_t	Rissöffnungsverschiebung an der Rissspitze	mm
δ_{tc}	kritische Öffnung	mm
Γ	geschlossener Integrationsweg	Nmm^{-2}
γ_t	Rissspitzenöffnungswinkel	rad
ν	Poissonzahl	–
Π	freigesetzte Energie beim Rissfortschritt	Nmm

Kapitel 1.

Einleitung

Im Kapitel 1 wird die Motivation zum Verfassen der vorliegenden Arbeit dargestellt. Nachfolgend im Abschnitt 1.2 wird die Problemstellung geschildert. Abschließend wird die Arbeit gegliedert.

1.1. Motivation

Numerische Simulationen sind, seit der raschen Entwicklung der Computertechnik und damit auch der Numerik in den 50er und 60er Jahren des 20. Jahrhunderts, in vielen technischen und wissenschaftlichen Bereichen weit verbreitet [5, 21, 23, 31]. Neben den analytischen und experimentellen Verfahren bilden die numerischen Verfahren die dritte und eine der wesentliche Säule bei der Untersuchung eines technisch-wissenschaftlichen Problems. In den Ingenieursdisziplinen hat sich die Finite-Elemente-Methode (FEM[1]) für die strukturmechanischen Berechnungen etabliert und gehört seit langer Zeit zum Standardwerkzeug des Berechnungsingenieurs [37, 59].

Neben einer gewissen Anzahl der kommerziellen Programmen existiert FEM-Software, die überwiegend in der Forschung eingesetzt wird. Meistens ist der Grund dafür der offene Quellcode. Somit lässt sich das Programm beliebig an eigene Probleme anpassen und erweitern [40]. Eines der bekanntesten Vertreter solch einer Software im Bereich der Strukturmechanik ist FEAP[2], das von Professor Robert Leroy Taylor[3] an der University of California, Berkeley in den 80er Jahren entwickelt wurde und seitdem permanent vom Professor Taylor und seinen Mitarbeitern verbessert wird.

Die Luft- und Raumfahrtindustrie fordert stets innovative Lösungen für vielseitige Problembereiche. Strukturmechanik stellt in diesem Zusammenhang keine Ausnahme dar und bietet ein breites Forschungsfeld an. Dabei sind die nicht-destruktiven Methoden zur Schadenanalyse von besonderem Interesse für die Entwickler der Luft- und Raumfahrttechnik. Aktive Strukturüberwachung ermöglicht es die Schäden rechtzeitig zu detektieren und im Anschluss zu beheben [24]. Allerdings sind solche Verfahren mit praktischem Aufwand wie Kabelbündel, die die Informationen von den Sensoren

[1] Die Finite-Elemente-Methode (englisch: finite element method) ist das am häufigsten routinemäßig eingesetzte Verfahren zur Berechnung komplexer Konstruktionen im Maschinenbau, im Apparatebau, in der Fahrzeugtechnik, in der Luft- und Raumfahrttechnik und im Bauwesen [37]

[2] Finite Element Analysis Program [61]

[3] Robert L. Taylor, amerikanischer Ingenieur und Professor am Department Bau- und Umweltingenieurwesen der University of California, Berkeley

übertragen, verbunden. Die Kabel bringen zusätzliches Gewicht, was den Treibstoffverbrauch der Flugzeuge bzw. Raumfahrtapparate erhöht. Um das Gewicht zu sparen bietet es sich an die Kabel durch geeignetere Leiter zu ersetzen, zum Beispiel durch die aus Silber gedruckte Leiterbahnen. Die gedruckte Elektronik kann nicht nur auf die Oberfläche, sondern auch in die Struktur integriert werden [28]. Demzufolge wird Silber zum tragenden Material und dessen bruchmechanische Parameter zu ermitteln sind. Dementsprechend ist es sinnvoll die Bruchvorgänge mit kohäsiven Zonen zu modellieren. Allerdings lassen sich die kohäsiven Parameter schwer experimentell ermitteln, so dass man zu der Kopplung zwischen den experimentellen und numerischen Methoden greifen muss.

1.2. Problembeschreibung und Gliederung der Arbeit

Problembeschreibung

Gewisse physikalische Parameter wie z.B. kohäsive Parameter in der Bruchmechanik sind schwer alle experimentell und analytisch oder numerisch teilweise gar nicht zu bestimmen. Aus diesem Grund sollen experimentell gewonnene Ergebnisse mit numerischen Methoden gekoppelt werden. Lassen sich die Parameter nicht mit direkten Methoden, beziehungsweise aufwendiger und langsamer als im Vergleich zur anderen Verfahren bestimmen, so werden diese von indirekten Beobachtungen der Größen gewonnen. In der Fachliteratur spricht man dann von inversen Problemen [43]. Folglich ist die Bestimmung der gesuchten Parameter der inversen Probleme als die inverse Parameteridentifikation bekannt.

Gegenstand dieser Arbeit ist die Entwicklung eines Algorithmus zur Ermittlung der physikalischen Materialparameter wie das Elastizitätsmodul mittels Iterationen mit dem FEM-Programm FEAP. Die Arbeit soll eine Grundlage für die Modellierung zeitaufwendig auf konventionellen Wegen zu ermittelnder Kohäsivparameter sein.

Gliederung

Im Kapitel 2 der Arbeit werden die für diese Arbeit relevanten Grundlagen der Bruchmechanik und Optimierung vorgestellt. Abschließend ist der letzte Kapitelabschnitt der Implementierung in FEAP gewidmet.

Im Fokus des Kapitels 3 steht die Entwicklung des eigentlichen Algorithmus. Es werden drei unterschiedliche Algorithmen mit ihren Vor- und Nachteilen vorgestellt. Als physikalisches Modell wurde eine Kragscheibe mit linear-elastischem Materialverhalten gewählt. Zu optimieren ist in diesem Modell das Elastizitätsmodul, das anhand der Kraft und Durchbiegung der Scheibe bestimmt wird. Am Anfang des Kapitels geht man auf die Implementierung der Fortran Subroutinen[4] in FEAP ein. Dabei werden möglichst detailliert alle Nuancen untersucht. Nachfolgend werden die Ergebnisse entsprechend validiert.

Letztendlich diskutiert man im Kapitel 4 die gewonnenen Ergebnisse.

[4] Eigene, in sich geschlossene Programmeinheit in Fortran [39]

Das finale Kapitel 5 beinhaltet die Zusammenfassung der gesamten Arbeit und den Ausblick für zukünftige Arbeiten.

Kapitel 2.

Grundlagen

In diesem Kapitel werden die erforderlichen Grundlagen für die vorliegende Masterarbeit erarbeitet. Der Abschnitt Grundlagen der Bruchmechanik beinhaltet relevante Konzepte aus dem Fachgebiet und geht schließlich wird man weitergeleitet zum Modell des Kohäsivzonenmodell. In dem Abschnitt 2.2 zu den Grundlagen der Optimierung sind die wichtigsten Grundideen der Optimierung zusammengefasst. Angefangen mit allgemeinen Grundlagen kommt man zu dem Kern des Abschnittes, der Methode der kleinsten Quadrate und den Optimierungsalgorithmen.

2.1. Grundlagen der Bruchmechanik

Aus oben genannten Gründen beschränkt man sich auf die wenigen ausgewählten Themen. Für einen tieferen Einblick in die Theorie empfiehlt sich z.B. Gross und Seelig [26], Rossmanith [50] oder Anderson [1].

2.1.1. Allgemeines

Die Ursachen für Materialbruch sind vielfältig und werden auf die mikroskopischen Effekte zurückgeführt. Diese Tatsache erschwert die Beschreibung des Bruches, weil die Eigenschaften des selben Materials auf der Mikroebene sehr unterschiedlich sind [26]. Für ingenieurtechnische Anwendungen und für das grundsätzliche Bruchverhalten eines unbekannten Materials bieten sich daher makromechanische Untersuchungen an. Diese werden in der vorliegenden Arbeit behandelt.

Ein makroskopischer Riss, der von einem nicht beschädigten Werkstoff umgeben ist, welcher mit den Ansätzen der Kontinuumsmechanik beschreibbar ist, hat an der Spitze sehr hohe und inhomogene Spannungen. Solche Spannungskonzentrationen können nicht mit den Konzepten der Festigkeitslehre beschrieben werden und müssen besonders behandelt werden [41]. In diesem Abschnitt werden die bekanntesten Konzepte der Bruchmechanik vorgestellt.

Die Bruchmechanik wurde in den 20er Jahren des 20. Jahrhunderts entwickelt und basiert auf der Theorie von Griffith, die den idealen spröden Bruch beschreibt [27]. Allerdings gelten die Annahmen von Griffith nicht für metallische Werkstoffe und sind deswegen nur begrenzt anwendbar. Aus diesem Grund mussten die Annahmen erweitert werden, was zu der linearen Bruchmechanik führte [27]. Die Annahmen des

linear-elastischen Verhaltens in LEBM[1] ist lediglich eine Idealisierung und verliert ihre Gültigkeit, wenn der Radius R der plastischen Zone (siehe Abbildung 2.3) nicht mehr klein ist [7]. Die Erweiterung der Theorie auf größere plastische Verformungen führt zu der elastisch-plastischen Bruchmechanik [7].

Die Abbildung 2.1 fasst die wichtigsten Aspekte der Bruchmechanik in einem Diagramm zusammen. Die wichtigsten Aspekte werden in kommenden Abschnitten dieses Kapitels näher erläutert.

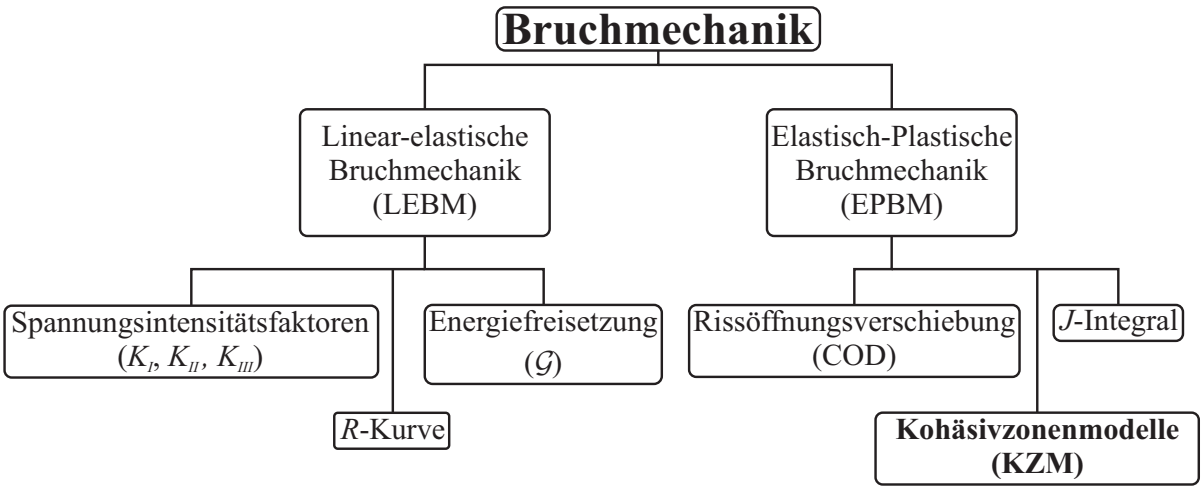


Abbildung 2.1.: Übersicht über die wichtigsten Konzepte der Bruchmechanik.

Vollständigkeitshalber sind weitere Konzepte nach Hahn [27] aufgelistet:

- Verzerrungsenergiedichte,
- Äquivalente Energiemethode,
- Nichtlineare Energiemethode,
- 2-Parameter-Konzept,
- COA (Rissöffnungswinkel),
- u.a.

Der, für die Arbeit, wichtigste Kernaspekt Kohäsivzonenmodell ist in der Abbildung 2.1 in fetter Schrift hervorgehoben und wird in dem Unterabschnitt 2.1.4 erklärt.

Bruchmoden

Betrachtet man die Bruchvorgänge genauer, so stellt man fest, dass man nur drei Arten des Bruches unterscheidet, die in der Fachliteratur als Bruchmoden bekannt sind. Alle andere Brucherscheinungen werden als die Superposition der drei grundlegenden Moden betrachtet und sind unter dem Namen *mixed-modes* bekannt [7, 41]. Die Abbildung 2.2 stellt schematisch die Moden dar.

[1] Linear-elastische Bruchmechanik

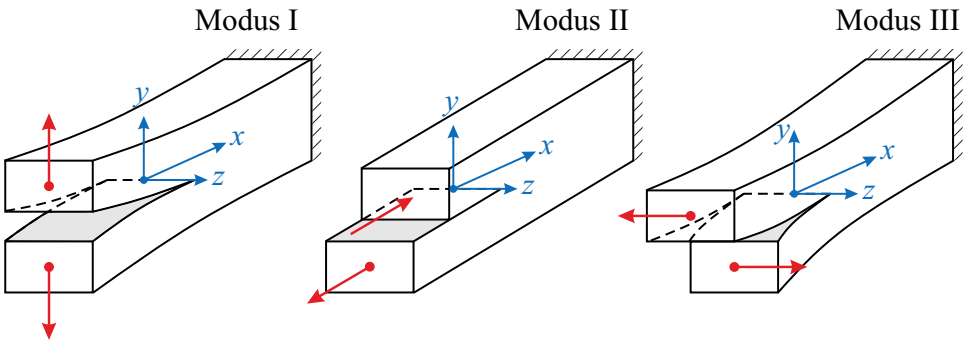


Abbildung 2.2.: Bruchmoden, nach [26].

Modus I stellt die Öffnung des Risses unter Normalspannungen dar. Dabei verschwindet die Scherspannung entlang der Rissufer und die Rissöffnung ist symmetrisch. Dieser Modus ist die gefährlichste Beanspruchungsart. Modus II zeigt eine ebene Scherbelastung in x-Richtung [50]. Letztlich wird durch Modus III die Trennung in z-Richtung dargestellt [26].

2.1.2. Linear-elastische Bruchmechanik (LEBM)

Ist die plastische Zone an der Rissspitze (siehe Abbildung 2.3) sehr klein, dass man im gesamten Gebiet von einem linear-elastischen Materialverhalten ausgeht, so spricht man von der linear-elastischen Bruchmechanik. Solche Lokalisierung des Bruchprozesses ist typisch für metallische Werkstoffe und die meisten spröden Materialien [26]. Die Abbildung 2.3 stellt die Umgebung der Rissspitze eines ebenen Körpers dar.

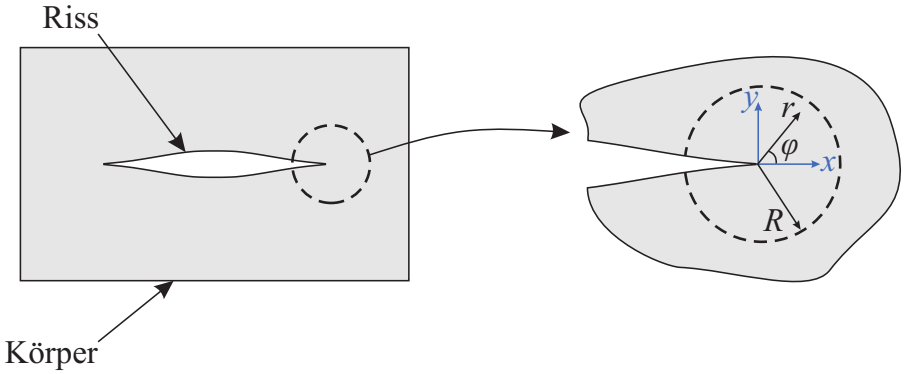


Abbildung 2.3.: Umgebung der Rissspitze, nach [26].

Dabei sind r und φ die Polarkoordinaten, die zur eleganteren Beschreibung des Spannungszustandes dienen. R ist der Radius der plastischen Zone in der nahen Umgebung der Rissspitze.

K-Konzept

Der Begriff K-Konzept ist in der Fachliteratur auch als Spannungsintensitätsfaktoren (SIF) bekannt und beschreibt die Spannungsverteilung im unmittelbaren Nahfeld der Rissspitze. Die Faktoren hängen von der Geometrie des Körpers, der Größe, der Lage

des Risses und der Belastungsart ab, d.h. $K_i = K_i$(Geometrie, Riss, Belastung, Material), $i = \{I, II, III\}$ [41]. Die analytischen Methoden liefern zwar Lösungen in geschlossener Form, allerdings sind sie nur für wenige einfache Probleme gegeben, so dass man bei komplexeren Problemen auf die Numerik angewiesen ist [26, 41]. Alternativ greift man zu den aufwendigeren experimentellen Methoden [26].

Erreicht der Spannungsintensitätsfaktor eine kritische Größe, die als Bruchzähigkeit K_c bezeichnet ist, so kommt es zum Bruch [26]. Somit liefert das K-Konzept folgendes Bruchkriterium :

$$K_I = K_{Ic}, \quad K_{II} = K_{IIc}, \quad K_{III} = K_{IIIc}. \tag{2.1}$$

Liegt ein *mixed-mode* vor, so dass alle drei Bruchmoden auftreten, gilt folgendes Bruchkriterium:

$$\left(\frac{K_I}{K_{Ic}}\right)^{\hat{u}} + \left(\frac{K_{II}}{K_{IIc}}\right)^{\hat{v}} + \left(\frac{K_{III}}{K_{IIIc}}\right)^{\hat{w}} = 1. \tag{2.2}$$

Die Bruchzähigkeit ist ein Materialkennwert, der für jeden Werkstoff experimentell bestimmt werden muss. Zusätzlich sind beim *mixed-mode* die Exponenten $\hat{u}$, $\hat{v}$, $\hat{w}$ zu ermitteln [7].

Energiefreisetzungsrate

Ein anderes Bruchkriterium ist die Energiefreisetzungsrate $\mathcal{G}$ bei der Rissausbreitung. Die Rissvergrößerung erfordert eine bestimmte Energie, die entweder als geleistete Arbeit der wirkenden Kraft oder durch die Freisetzung der gespeicherten Energie $-\Pi$ bei einem verformten Bauteil gegeben ist [7]. Betrachtet man einen infinitesimalen Rissfortschritt dA und die darauf bezogene freigesetzte Energie $-\mathrm{d}\Pi$, so ist die Energiefreisetzungsrate folgendermaßen definiert [26]:

$$\mathcal{G} = -\frac{\mathrm{d}\Pi}{\mathrm{d}A}, \tag{2.3}$$

bzw. bei einem ebenen Problem bezieht man sich auf die infinitesimale Risslänge da:

$$\mathcal{G} = -\frac{\mathrm{d}\Pi}{\mathrm{d}a}. \tag{2.4}$$

In einem linear elastischen Fall besteht zwischen der Energiefreisetzungsrate und den Spannungsintensitätsfaktoren ein direkter Zusammenhang, der folgende Gestalt aufweist [26]:
Modus I:

$$\mathcal{G} = \begin{cases} \dfrac{K_I^2}{E} & \text{ESZ}^2, \\ \dfrac{(1-\nu^2)K_I^2}{E} & \text{EVZ}^3. \end{cases} \tag{2.5}$$

[2] ebener Spannungszustand
[3] ebener Verzerrungszustand

Modus II:

$$\mathcal{G} = \begin{cases} \dfrac{K_{II}^2}{E} & \text{ESZ}, \\ \dfrac{(1-\nu^2)K_{II}^2}{E} & \text{EVZ}\,. \end{cases} \tag{2.6}$$

Modus III:

$$\mathcal{G} = \frac{1}{2G} K_{III}^2 \tag{2.7}$$

Für *mixed-mode* werden alle drei K-Faktoren herangezogen [26]:

$$\mathcal{G} = \begin{cases} \dfrac{1-\nu^2}{E}\left(K_I^2 + K_{II}^2\right) + \dfrac{1}{2G} K_{III}^2 & \text{ESZ}, \\ E\left(K_I^2 + K_{II}^2\right) + \dfrac{1}{2G} K_{III}^2 & \text{EVZ}\,. \end{cases} \tag{2.8}$$

Der kritische Wert der Energiefreisetzungsrate ist das Bruchkriterium und wird als Risswiderstand oder die Rissausbreitungsenergie genannt [7, 26]:

$$\mathcal{G} = \mathcal{G}_c. \tag{2.9}$$

Ähnlich der Bruchzähigkeit ist der Risswiderstand ebenfalls ein Materialkennwert, der experimentell ermittelt wird. Gross bezeichnet in [26] die Rissausbreitungsenergie auch als Bruchzähigkeit, weil zwischen den beiden Größen ein unmittelbarer Zusammenhang besteht.

2.1.3. Elastisch-plastische Bruchmechanik (EPBM)

Ist die plastische Zone des duktilen Werkstoffes an der unmittelbaren Rissspitze (siehe Abbildung 2.3) nicht mehr klein, so spricht man von elastisch-plastischer Bruchmechanik oder von der Fließbruchmechanik [7]. Die Annahmen der linear-elastischen Bruchmechanik gelten nicht mehr und müssen entsprechend erweitert oder gar ersetzt werden. Die Plastifizierung führt beim Erhöhen der Belastung zum Abstumpfen der Rissspitze, was die plastische Zone erweitert [26, 41]. Die beiden wichtigsten Konzepte der EPBM sind die Rissöffnungsverschiebung (CTOD[4]) und das J-Integral, worauf in folgenden Unterabschnitten eingegangen wird. Hierbei ist das CTOD-Konzept im Vergleich zum J-Integral eher experimentell [26].

Rissöffnungsverschiebung (CTOD)

Die Rissöffnungsverschiebung basiert auf der Annahme, dass der Bruch von der plastischen Verformung an der Rissspitze und nicht mehr von den Spannungsintensitätsfaktoren dominiert wird [7]. Mit steigender Belastung stumpft die spitze Rissspitze vor dem Eintreffen des eigentlichen Bruches kontinuierlich ab [41], wie es in der linken

[4] Crack Tip Opening Displacement

Abbildung 2.4 dargestellt ist. Ein Maß für die plastische Verformung ist die Rissöffnungsverschiebung (CTOD) δ_t an der Rissspitze. Die Berechnung von δ_t erfolgt durch Konstruktion von zwei Sekanten unter 45° in der abgestumpften Spitze [7, 41]. Eine andere Möglichkeit zur Bruchanalyse bei dem vorliegenden Konzept ist die Benutzung des Rissspitzenöffnungswinkels (CTOA[5]) γ_t. Die graphische Darstellung ist in der rechten Abbildung 2.4 dargestellt. Allerdings ist CTOA aufgrund schwer durchführbaren Messungen nicht so weit verbreitet [7]. Der interessierte Leser kann sich einen tieferen Einblick zu dem Rissspitzenöffnungswinkel z.B. bei Darcis [15] oder Sakhalkar [51] verschaffen.

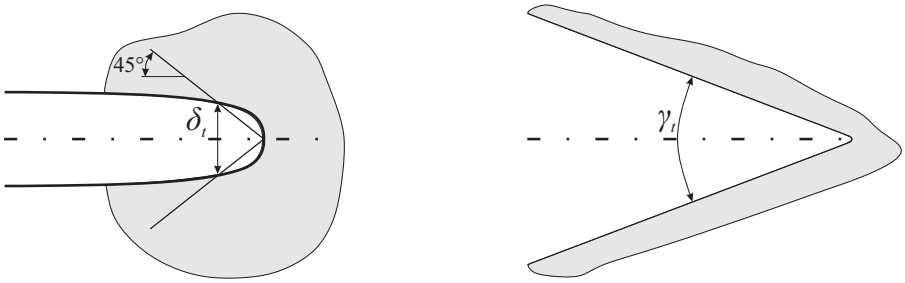


Abbildung 2.4.: *links:* CTOD δ_t am ruhenden Riss, *rechts*: CTOA γ_t am bewegten Riss, nach [41].

Als Bruchkriterium wurde von Buedekin & Stone [9] und Wells [58] vorgeschlagen, dass der Bruch beim Erreichen der kritischen Öffnung δ_{tc} eintrifft. Es gilt also folgende Beziehung:

$$\delta_t = \delta_{tc} \,. \tag{2.10}$$

Die kritische Öffnung ist ein materialspezifischer Wert [41].

Die experimentelle Ermittlung von CTOD erweist sich als kompliziert, weil man die abgestumpfte Rissspitze schwer erreicht. Aus dem Grund wird die Rissöffnung (COD[6]) an der Oberfläche gemessen und mit Hilfe geometrischen Annahmen auf die Rissspitze extrapoliert [41].

J-Integral

Die Berechnung des Spannungs- und Verzerrungszustandes in der Umgebung der Rissspitze bei der EPBM ist ein komplexes Problem [50]. Eine wichtige Methode zur Definition eines Bruchkriteriums der Fließbruchmechanik ist das von Cherepanov [11] und Rice [49] eingeführte *J*-Integral. Es ist ein Linienintegral mit einem geschlossenen Integrationsweg um die Rissspitze [7] und spielt eine vergleichbare Rolle in der EPBM wie die Spannungsintensitätsfaktoren in der LEBM [41]. Die Abbildung 2.5 zeigt schematisch die Grundidee hinter der Definition des Konzeptes.

[5]Crack Tip Opening Angle
[6]Crack Opening displacement

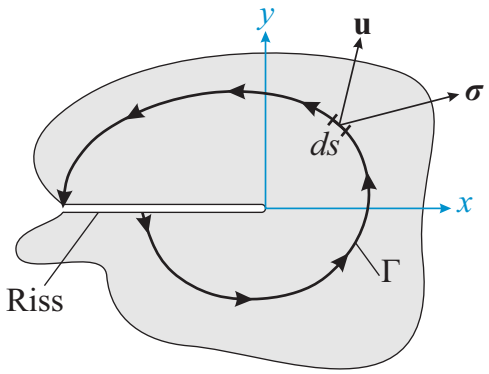


Abbildung 2.5.: *J*-Integral, nach [7].

In der Abbildung vorkommende Größen sind:

- $\boldsymbol{\sigma}$ Spannungsvektor
- $\mathbf{u}$ Verschiebungsvektor
- ds Wegelement
- Γ geschlossener Integrationsweg.

Das Integral ist folgendermaßen definiert:

$$J = \int_{\Gamma} \left(W_e \, dy - \boldsymbol{\sigma} \frac{d\mathbf{u}}{dx} ds \right). \tag{2.11}$$

Dabei ist

$$W_e = \int_0^{\varepsilon} \sigma_{ij} \, d\varepsilon_{ij} \, ,$$

die Dehnungsenergiedichte oder auch die auf das Volumen bezogene Formänderungsarbeit [7].

Aus dem Energieerhaltungssatz folgt, dass beim stetigen Integrationsweg das *J*-Integral zu Null wird. Wiederum ist das Integral ungleich Null im Falle einer Singularität des Risses, welches allerdings unabhängig vom Integrationsweg. Daraus folgt, dass der Integrationsweg entsprechend weit von der Rissspitze gewählt werden darf, was die Berechnung deutlich erleichtert [7]. Experimentelle Bestimmung des *J*-Integrals, z.B. nach Frediani [18], ist im Allgemeinen aufwendig [50].

Ähnlich wie in vorherigen Konzepten ist das Bruchkriterium beim Erreichen des kritischen materialspezifischen Wertes J_c definiert [41]:

$$J = J_c \, . \tag{2.12}$$

Die kritische Größe wird in genormten Experimenten bestimmt [26].

2.1.4. Kohäsivzonenmodell

Der Bruch bei den Kohäsivzonenmodellen (KZM) findet ausschließlich in einem schmalen und streifenförmigen Bereich statt. Der restliche Körper bleibt dabei schädigungsfrei [41]. Verbreitete Anwendung finden die KZM vor allem bei der Berechnung von Schädigungen in Faserverbundstrukturen. Beispielshalber ist man bei der Auslegung der geklebten Spante und Stringer in Flugzeugrümpfen auf die Modellierung der Klebschichten mit den Kohäsivzonenmodellen angewiesen. Weitere Anwendungsgebiete sind die Modellierung von Keramiken und Beton im Bauingenieurwesen und Modellierung vom duktilen Bruch und engspaltigen Schweißnähten im allgemeinen Maschinenbau [41].

Die ersten Arbeiten zu den Kohäsivzonenmodellen wurden in den früheren 60er Jahren des 20. Jahrhundert von Barenblatt [4] und Dugdale [16] veröffentlicht. Obwohl das Modell von Barenblatt keine Vorteile im Vergleich zu dem *K*-Konzept bringt [26], wird es an der Stelle zur Verdeutlichung des KZM vorgestellt. Abbildung 2.6 zeigt den schematischen Aufbau der Kohäsionszone.

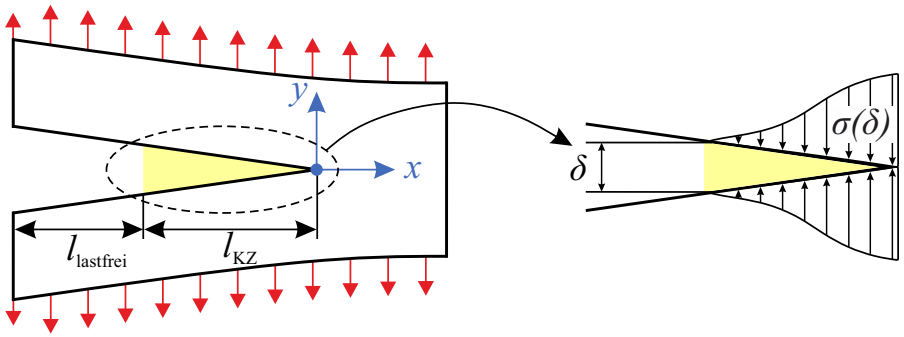


Abbildung 2.6.: Kohäsionszone.

Über die Länge l_{KZ} wirken intermolekulare Kohäsionsspannungen $\sigma(\delta)$, die von der Öffnung δ abhängig sind.

Physikalisch lässt sich das Modell wie, in der Abbildung 2.7 gezeigt ist, darstellen. Im oberen Bildbereich ist das physikalische Modell mit Materialschäden und einem initiierten Riss zu sehen. Man idealisiert das Konzept und ersetzt das Material durch die Kohäsivelemente. Nach Überschreiten der kritischen Kohäsionsspannung werden die Elemente entsprechend gebrochen.

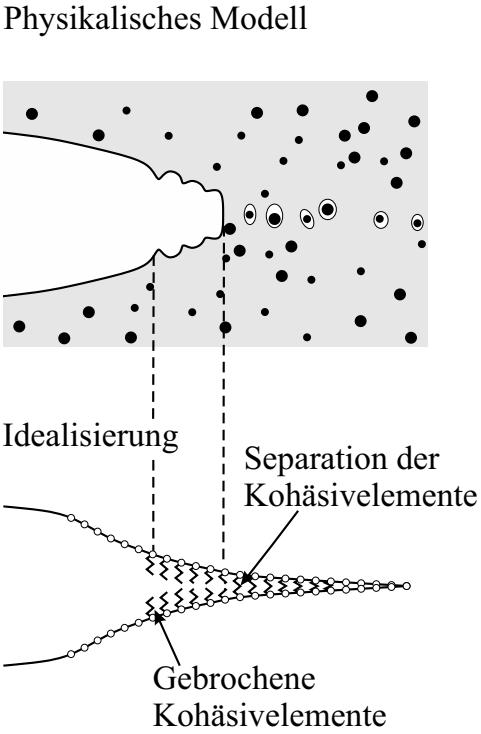


Abbildung 2.7.: Physikalische Darstellung des Kohäsivzonenmodells, nach [12].

Die Wechselwirkung zwischen der beiden Rissufern wird bei KZM durch die sogenannte Separationsgesetze (TSL[7]) beschrieben, auf die im darauffolgenden Unterabschnitt näher eingegangen wird.

Noch ist das Modell für Rissausbreitungen in einem Werkstoff determiniert. Allerdings verwendet man KZM seit den letzten Jahrzehnten für die Festigkeitsvorhersagen von Klebeverbindungen, als ein Zusatz zu FE-Analysen, die eine Simulation des Schadenwachstums von kontinuierlichen Materialien oder Schnittstellen zwischen verschiedenen Materialien ermöglicht [10, 13]. Die ersten FEM Implementierungen der KZM sind auf Hillerborg [30] zurückzuführen. Numerische Simulationen der Klebeverbindungen sind komplex, weil deren Verhalten von adhäsiven Eigenschaften und Grenzschichten abhängig sind [34].

Betrachtet man die Bruchkörper, so stellt man fest, dass einige von denen sowohl adhäsives als auch kohäsives Versagensverhalten ausweisen. In der Abbildung 2.8 wird schematisch der Unterschied zwischen dem kohäsiven und adhäsiven Bruch anhand des DCB[8]-Versuchs dargestellt. Die Schicht des Verbindungswerkstoffs ist im Bild übermäßig dargestellt und soll lediglich der Anschauung dienen. Kombiniert man die numerischen oder analytischen Berechnungen der KZM mit den experimentell ermitelten Werten, so muss sicher gestellt werden, dass die Versuchsproben auch ein kohäsives Verhalten aufweisen. Ansonsten begeht man die Gefahr unsinnige Vergleiche zu machen.

[7] Traction Separation Laws
[8] Double Cantilever Beam

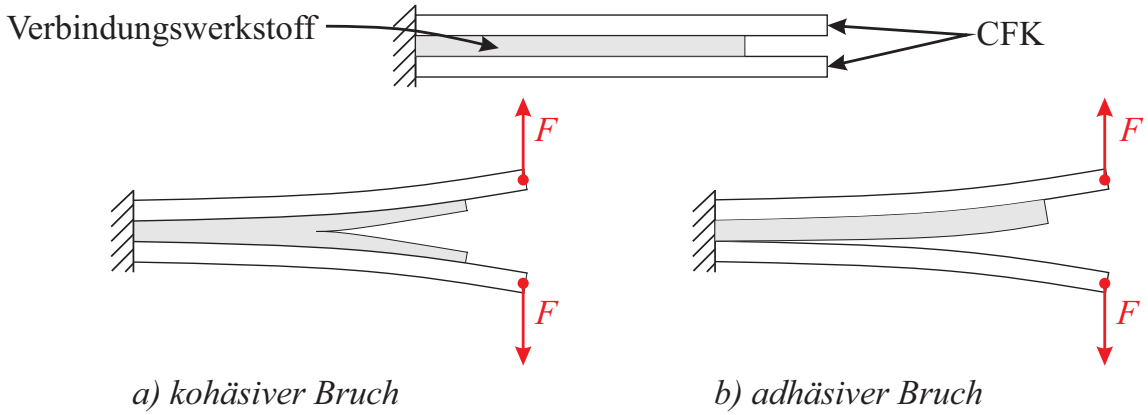


Abbildung 2.8.: Unterschied zwischen kohäsivem und adhäsivem Bruch am Beispiel des DCB-Tests, bzw. des Mode I.

Die in der Abbildung 2.8 gezeigten CFK[9]-Platten folgen den Gesetzen der Kontinuumsmechanik und die Schicht des Verbindungswerkstoffs dem Separationsgesetz.

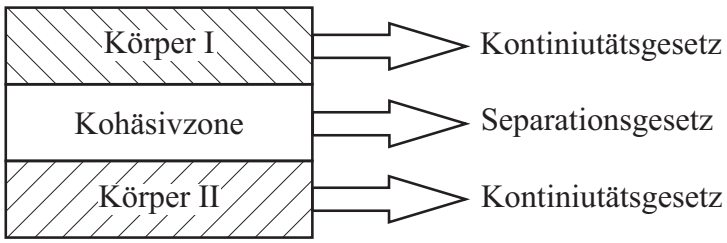


Abbildung 2.9.: Physikalische Gesetze des DCB.

Für weiterführende Informationen sei der interessierte Leser auf die einschlägige Literatur wie Elices [17], Hui [32], Mahler [44] oder Schwalbe [53] verwiesen.

[9]Carbonfaserverstärkter Kunststoff

Separationsgesetze

Das Separationsgesetz stellt den Zusammenhang zwischen der Randspannung $\sigma(\delta)$ und der Separation der Grenzflächen δ (siehe die Abbildung 2.6) dar. Es existieren diverse TSL, die sich je nach Werkstoff in ihrer Form unterscheiden [41]. Dabei haben die Separationsgesetze einen starken Einfluss auf die Ergebnisse. Die Wahl der Kohäsivparameter hängt von der Form der Funktion [52]. Die Abbildung 2.10 zeigt ein typisches bi-lineares Separationsgesetz. Mit steigender Rissöffnung steigt die Kohäsionspannung bis sie den kritischen Materialwert σ_c bei der Separation δ_0 erreicht. Daraufhin erfolgt die Trennung des Werkstoffs. Die Rissöffnung wächst bis zu dem kritischen Wert δ_c an. Danach ist das Material getrennt und es können keine Spannungen mehr übertragen werden [41]. Zu der maximalen Spannung σ_c gehört die Separation δ_0.

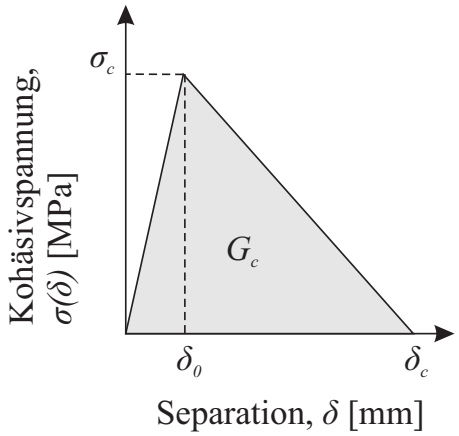


Abbildung 2.10.: Separationsgesetz, nach [60].

Integriert man die Funktion des Separationsgesetzes von Null bis δ_c, so erhält man die Seaprationsenrgie:

$$G_c = \int_0^{\delta_c} \sigma(\delta)\, d\delta, \tag{2.13}$$

die als die Fläche unter der Kurve zu interpretieren ist. Verhält sich der umgebende Werkstoff elastisch, so entspricht die Separationsenergie der Energiefreisetzungsrate und dem *J*-Integral [26].

Wie zum Anfang des Unterabschnittes erwähnt existieren viele unterschiedliche Ansätze für die Separationsgesetze. Die Abbildung 2.11 fasst die typischen TSL zusammen.

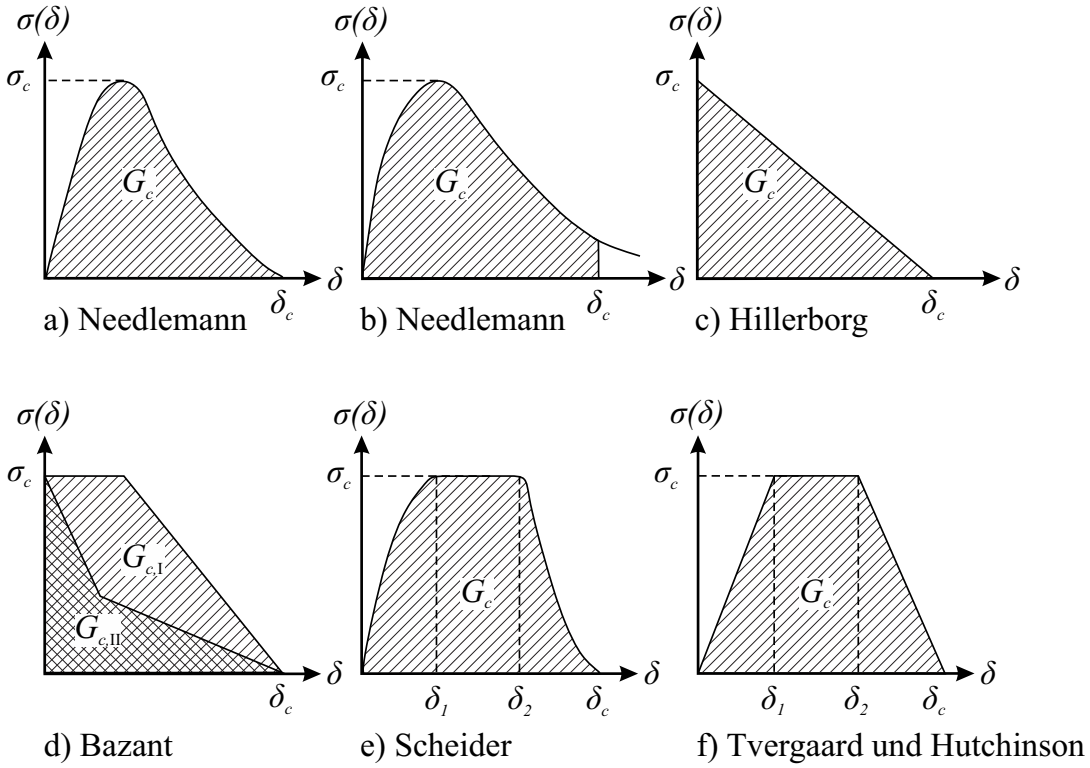


Abbildung 2.11.: Typische Separationsgesetze , nach [53].

Die TSL sind lediglich Annahmen, die von dem Anwender benutzt werden, um gewisse Materialverhalten zu beschreiben. Meistens sind die Parameter nicht bekannt und müssen experimentell werden. Z.B. nach KUNA [41] lassen sich die Parameter δ_c, σ_c und G_c folgendermaßen bestimmen:

- δ_c aus der Vermessung der Bruchprozesszone,
- σ_c aus der Bruchspannung von glatten oder gekerbten Zugproben,
- G_c aus Bruchmechanikversuchen über K_{Ic} oder J_i .

Zum einen ist die experimentelle Ermittlung aller Parameter im Allgemeinen nicht möglich und es gibt keine genormte Regelung zum Durchführen der Versuche [60], zum anderen ist seit der intensiven Entwicklung der FE-Methoden die Interesse zur rein experimentellen Ermittlung der kohäsiven Gesetze vergleichsweise gesunken [55]. Während die Entwicklung numerischer Methoden intensiv anwuchs, erwecken die Methoden zur experimentellen Messung der kohäsiven Gesetze ein vergleichsweise kleineres Interesse [7]. Eine Möglichkeit zum Ermitteln der Parameter ist die inverse Parameteridentifikation. Der Sinn der Methode ist die Bestimmung der Parameter aus den einfachen Kraft-Verschiebungsmessungen. Nach MAIER U.A. [45] sind folgende Schritte nötig:

- Datenerfassung aus experimentellen Tests im Labor,
- Computer-Simulation der Tests, in denen man das Modell kalibriert,

- Minimierung der Residuen[10], die die Differenz zwischen den experimentellen Daten und den entsprechenden simulierten Werten in Bezug auf die gesuchten Parameter darstellen.

Der letzte Punkt der Auflistung weist auf ein Optimierungsproblem hin.

2.2. Grundlagen der Optimierung

Allgemein lässt sich ein Optimierungsproblem, wie in der Abbildung 2.12 dargestellt, lösen. Von dem Anwender ist es notwendig das Problem zu formulieren und einen Startvektor festzulegen. Die tatsächliche Berechnung soll in dem Computer stattfinden. Schließlich müssen die Ergebnisse sorgfältig analysiert werden.

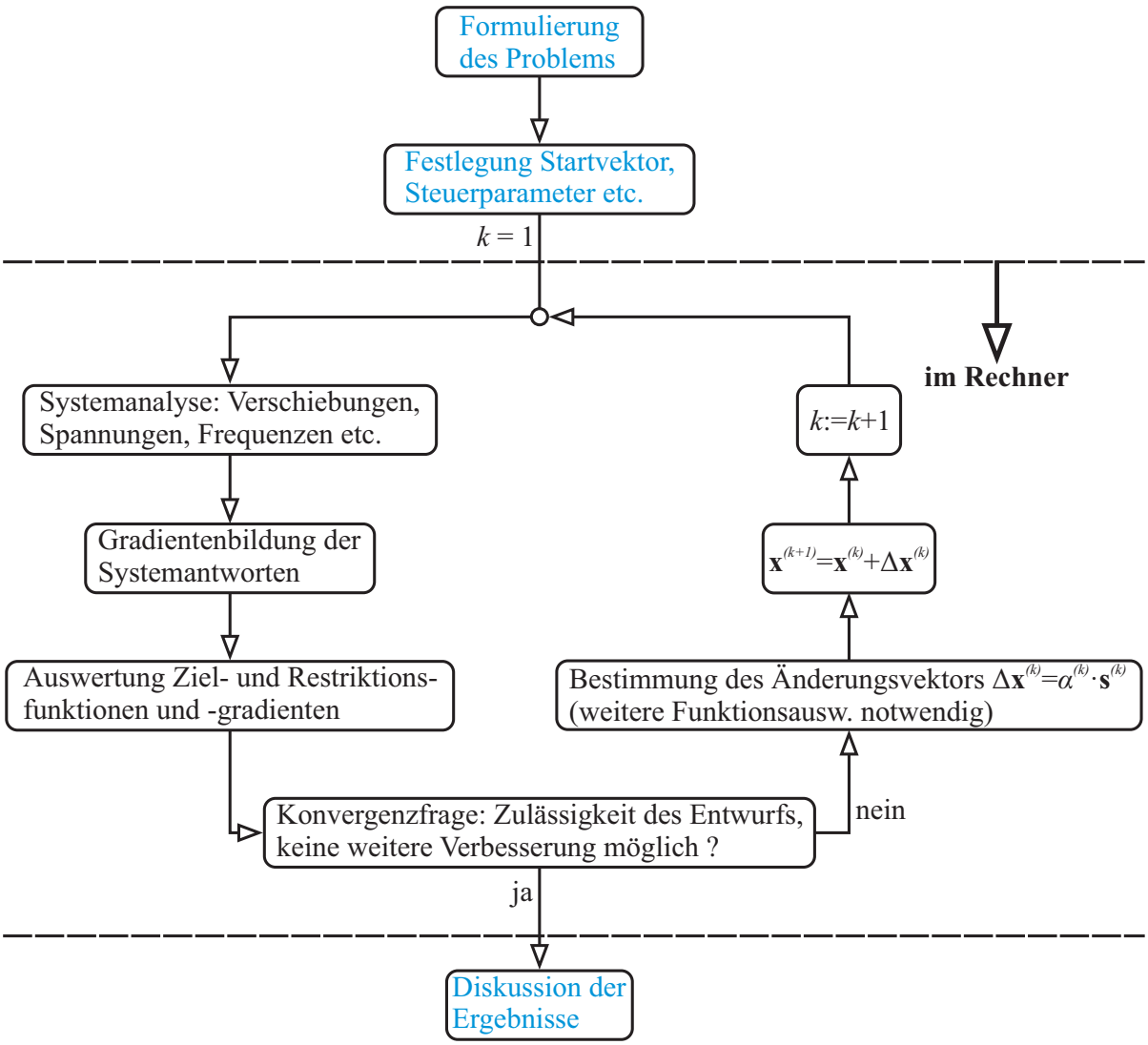


Abbildung 2.12.: Optimierungsprinzip, nach [3].

Die Bestimmung der Parameter wird von numerischen Algorithmen übernommen. Eine genaue Übersicht der Algorithmen zur numerischen Parameteridentifikation sind

[10]Differenzen zwischen Daten und Modell [6]

in Apel [2] gegeben. Der Autor unterteilt die Optimierung in restringiert und unrestringiert Optimierungsprobleme. Für die vorliegende Arbeit sind nur die unrestringiert Probleme von Relevanz, weil man vorerst keine Einschränkungen an die Parameter hat. Der Interessierte Leser sei an der Stelle auf die Fachliteratur verwiesen. Numerische Optimierungstechniken sind z.B. in Nocedal [47] zu finden. Mathematische Formulierungen der Optimierung sind in Geiger [22] und in Jarre [33] beinhaltet.

2.2.1. Methode der kleinsten Quadrate

In manchen technisch-wissenschaftlichen Disziplinen ist es notwendig unbekannte Parameter einer Funktion, die beispielsweise ein spezifisches Materialverhalten beschreiben, jedoch oft nicht bekannt sind, zu quantifizieren. Sie müssen daher aus der Reihe von Messungen ermittelt werden. Die Anzahl der vorgenommenen Messungen ist in der Regel größer als die Zahl der unbekannten Parameter [14, 54]. Somit ist das System nicht exakt lösbar und man muss auf statistische Techniken zurückgreifen. Eine einfache im Sinne der Benutzung und dennoch sehr effektive Vorgehensweise die Parameter zu bestimmen ist die Methode der kleinsten Quadrate (MKQ) [42]. Sie geht auf die Arbeiten von Gauß, Laplace, und Legendre ins 19. Jahrhundert zurück [6, 19].

Das Prinzip der MKQ ist die Minimierung der Quadrate der Residuen. Die Abbildung 2.13 verdeutlicht das Prinzip.

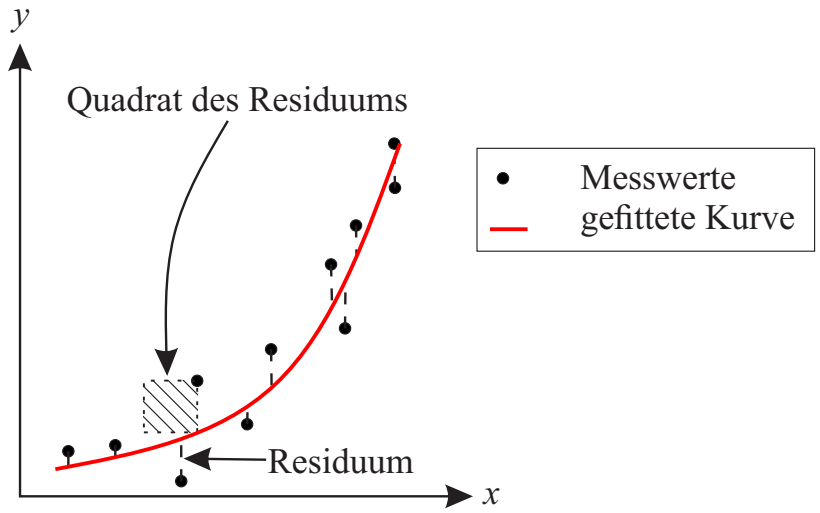


Abbildung 2.13.: Prinzip der Methode der kleinsten Quadrate.

Die Residuen sind in (2.14) definiert:

$$r_i = \sum_{i=1}^{n} y_i - \varphi_i(x_1, \ldots, x_m), \qquad i = \{1, 2, \ldots, n\}, \qquad m, n \in \mathbb{N}. \tag{2.14}$$

Dabei sind $y_i \in \mathbb{R}^n$ die Messwerte und $\varphi_i(x_1, \ldots, x_m) \in \mathbb{R}^m$ die Funktion der m Parameter. Man fordert an der Stelle das Minimum der Quadrate der Residuen:

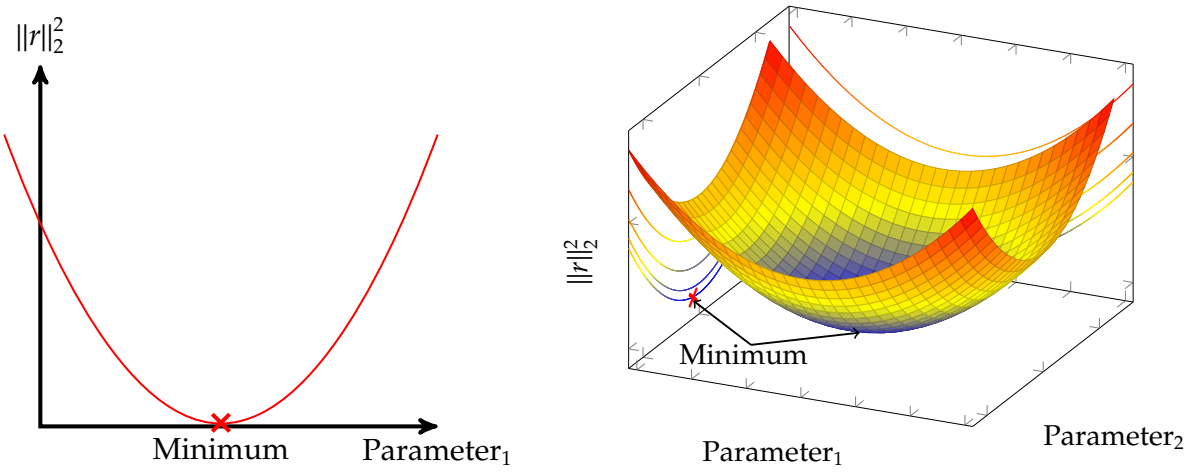
$$S = r_i^2 = \sum_{i=1}^{n} (y_i - \varphi_i(x_1, \ldots, x_m))^2 = \min . \tag{2.15}$$

Somit minimiert man die Euklidische Norm des Fehlers:

$$\|r_i\|_2^2 = \mathbf{r}^T \mathbf{r} = \min, \tag{2.16}$$

wobei $\mathbf{r}$ der Residuenvektor und $\|\ldots\|_2$ die euklidische Norm ist.

Grafisch lässt sich die Gleichung 2.14 am Beispiel eines Systems mit dem quadratischen Verlauf von S folgend darstellen:



(a) Ein Parameter spannt einen $\mathbb{R}^2$-Raum. (b) Zwei Parameter spannen einen $\mathbb{R}^3$-Raum.

Abbildung 2.14.: Grafische Darstellung der Summe der Quadrate.

Wie man der Abbildung 2.14 entnehmen kann, spannt ein Parameter einen ebenen Raum auf, in dem das Minimum zu finden ist. Identifikation nur eines Parameters spannt einen $\mathbb{R}^2$ Suchraum auf (Abbildung 2.14(a)) und beim Hinzufügen eines weiteren Parameters erhöht sich die Dimension des Raumes um eins (Abbildung 2.14(b)). Hat man n Parameter zu identifizieren, so wird nach folgender Vorschrift

$$n \rightarrow \mathbb{R}^{n+1}, \qquad n \in \mathbb{N}^*,$$

ein Suchraum erzeugt, der zwar nur für $n \leq 2$ grafisch darstellbar ist. Im anderen Fall gelten dennoch die Regeln der Optimierung.

Lineare kleinste Quadrate

Lineare kleinste Quadrate oder das lineare Ausgleichsproblem liegt vor, wenn $\varphi_i(x_1, \ldots, x_m)$ lineare Funktionen der x_j, $j = \{1, \ldots, m\}$ sind [19]. Es gibt dann eine Matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$

mit

$$\begin{bmatrix} y_1 - \varphi_1(x_1, \ldots, x_m) \\ \vdots \\ y_n - \varphi_n(x_1, \ldots, x_m) \end{bmatrix} = \mathbf{y} - \mathbf{A}\mathbf{x}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} x_1 \\ \vdots \\ y_n \end{bmatrix}. \tag{2.17}$$

Man minimiert entsprechend die Norm:

$$\min_{\mathbf{x} \in \mathbb{R}^m} \left\| \mathbf{y} - \mathbf{A}\mathbf{x} \right\|_2 . \tag{2.18}$$

Es ist also ein Vektor $\mathbf{x} \in \mathbb{R}^m$ gesucht, der folgende Funktion minimiert:

$$\left\| \mathbf{y} - \mathbf{A}\mathbf{x} \right\|_2^2 = (\mathbf{y} - \mathbf{A}\mathbf{x})^T (\mathbf{y} - \mathbf{A}\mathbf{x}). \tag{2.19}$$

Das Optimum existiert, wenn die Bedingung:

$$\nabla_{\mathbf{x}} (\mathbf{y} - \mathbf{A}\mathbf{x})^T (\mathbf{y} - \mathbf{A}\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{y} = 0, \tag{2.20}$$

erfüllt ist. Dabei ist $\nabla_{\mathbf{x}} = \dfrac{\partial(\ldots)}{\partial x_i}$. Die Gleichung (2.20) umgeformt ergibt:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}. \tag{2.21}$$

Die linearen Gleichungen (2.20) bzw. (2.21) werden als die Normalengleichungen genannt. Ist die Matrix $\mathbf{A}$ nichtsingulär und positiv definit, so ist die Lösung der Normalengleichungen auf diese Weise definiert [19]:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{2.22}$$

Ein wichtiges Kriterium für die Lösbarkeit der Gleichung (2.22) ist die Kondition des Terms ($\mathbf{A}^T\mathbf{A}$) bzw. der Matrix[11] $\mathbf{A}$. Ist die Konditionszahl klein und zwar nur wenig größer als 1, so ist das Problem gut konditioniert [56]. Ist die Konditionszahl groß, so ist das Problem schlecht konditioniert und man hat bei kleinen Störungen von $\mathbf{A}$, wie z.B. numerische Ungenauigkeiten, großen relativen Fehler des Lösungsvektors $\mathbf{x}$ [19]. Entsprechend müssen zur Lösung des Problems (2.22) numerische Methoden wie Cholesky-, QR- oder Singulärwertzerlegung herangezogen werden [14].

Nichtlineare kleinste Quadrate

Im Gegensatz zu den linearen Ausgleichsproblemen besteht jetzt zwischen $\varphi_i(x_1, \ldots, x_m)$ und x_j nichtlinear voneinander ab. In so einem Fall hat S keine quadratische Form. Direkte Berechnungsmethoden wie bei linearen kleinsten Quadraten können jetzt nicht angewandt werden und man muss auf die iterative Methoden zurückgreifen [6].

Man unterscheidet generell zwischen ableitungsbehafteten und ableitungsfreien Verfahren zur Lösung nichtlinearer Problemen [2]. Ableitungsbehaftete Berechnungstechniken nutzen die erste (Gradientenverfahren bzw. die Methode des steilsten Abstiegs) oder die zweite (Newton-Algorithmus, Gauß- Newton- Algorithmus, Levenberg-

[11] Da die Matrix nichtquadratisch ist, muss die spektrale Kondition gebildet werden [14]

Marquardt- Verfahren) Ableitungen der Zielfunktion um sie zu minimieren [3]. Wenn bei den direkten Verfahren die Kondition des Problems eine entscheidende Rolle spielt, so ist bei den iterativen Methoden die Konvergenz der Maß zur Beurteilung des Verfahrens.

Die *Methode des steilsten Abstieg* benötigt zwar nur die erste Ableitung, aber ist oft sehr langsam [2]. Man weiß auch nicht, ob weitere lokale Minima existieren [3].

Das *Newton-Verfahren* beruht auf einer quadratischen Approximation der Zielfunktion in jedem Iterationsschritt. Die Verwendung der Ableitung 2. Ordnung sorgt für eine sehr schnelle lokale Konvergenz. Allerdings muss die Hesse-Matrix[12] nichtsingulär und positiv definit sein [2, 3].

Offensichtlich stellt die Hesse-Matrix gewisse numerische Anforderungen an das Newton-Verfahren. Um dies zu vermeiden, wird bei dem *Gauß-Newton-Verfahren* diese Matrix durch $\mathbf{A}^T\mathbf{A}$-Matrix[13] approximiert [2, 25].

Levenberg-Marquardt-Verfahren stellt eine Kombination der Suchrichtungen aus dem Gradientenverfahren und dem Newton-Verfahren dar und eignet sich gut für nichtlineare Fehlerquadratprobleme [46, 47]. Der Algorithmus ist schnell und robust [57].

Die Ableitungsfreien Verfahren bestimmen das Minimum ausschließlich mit Hilfe des aktuellen Funktionswertes ohne auf die Ableitungen zuzugreifen. Vorteile bringen solche Verfahren, wenn die Zielfunktion nicht stetig oder nicht differenzierbar ist [3]. Die Algorithmen sind vielfältig und unterscheiden sich, im Gegensatz zu den ableitungsbehafteten Verfahren, sehr stark in ihren Ansätzen. An der Stelle werden die zwei wohl am meisten verbreiteten Algorithmen kurz vorgestellt.

Evolutionsstrategien sind Zufallsstrategien, die die biologische Prozesse der Evolution nachahmen und auf den Prinzipien Mutation und Selektion beruhen [2, 3]. Die Methoden sind extrem robust, weil sie gut mit fehlerhaften, ungenauen und unvollständigen Daten umgehen können. Nachteilig ist, dass die Evolutionsalgorithmen hohe Rechnerkapazitäten und Rechenzeiten erfordern. Zusätzlich ist es nicht sicher, dass die gefundene Lösung auch das globale Optima ist [38].

Beim *Monte-Carlo-Verfahren* werden die Suchpunkte mit Hilfe eines Zufallszahlengenerators gesetzt und die Zielfunktion wird an den Punkten ausgewertet. Der Algorithmus ist zwar einfach zu implementieren, benötigt allerdings hohe Rechen- und Zeitressourcen [3].

Die vorgestellten Algorithmen sind in ihren Ansätzen sehr unterschiedlich und man muss als Anwender die Verfahren an eigene Probleme anpassen. So können die ableitungsbehafteten Algorithmen auf die Problemstellungen angewandt werden, wenn die Zielfunktion bekannt ist. Hat man z.B. nur diskrete Punkte aus FE-Simulationen, so müssen in zusätzlichen Berechnungsschritten weitere Informationen zur Anwendung erarbeitet werden. Was die Effizienz der Verfahren deutlich verschlechtert.

[12] Matrix der zweiten Ableitungen
[13] siehe Lineare kleinste Quadrate

2.3. Allgemeines zur Implementierung in FEAP

Wie in dem Abschnitt 1.1 erwähnt ist FEAP ein Open Source FEM Programm, das dem Anwender erlaubt eigene Implementierungen in den Quellcode einzubinden. FEAP ist in FORTRAN 77 implementiert, was starke Einschränkungen wie ausschließlich statische Allokation oder veraltete Kontrollstrukturen wie `GOTO`-Anweisung mit sich bringt [39]. Arbeitet man aber mit *Intel Fortran Compiler* und *Microsoft Visual Studio*, so erlaubt der Compiler die eigenen Subroutinen in modernerer Fortran 90 Sprache zu schreiben. Auf fundamentalen Aufbau und Benutzung des Programms wird in dem Abschnitt nicht eingegangen. Dazu sei verwiesen auf die FEAP Handbücher[14] und die Werke von Taylor und Zienkiewicz [61, 62, 63].

Anders als bei der Implementierung eigener Elemente[15], gibt es in FEAP keine vordefinierte Schnittstellen zum einbinden von eigenen Algorithmen. Das Problem wird gelöst, in dem man eigene Subroutinen in der Datei `pcontr.f` folgendermaßen einbindet:

```
  ...
  elseif(pcomp(titl(1:4),'popt',4)) then
  call acheck(titl,yyy,20,80,160)
  read(yyy,"(A4,16X,A4,16X,A4,16X,A4,16X,A8,12X,A)",
&         err=900,end=911)
&         titl(1:4),dnam,dnam2,dnam3,dnam4,dnam5
  call popt(dnam,dnam2,dnam3,trim(dnam4),trim(adjustl(dnam5)))
  ...
```

Code 2.1: FEAP-Schnittstelle für eigene Algorithmen.

In der Subroutine `popt` ist der eigene Algorithmus implementiert. Wie die Implementierung im Detail aussieht, wird im Kapitel 3 erläutert.

Da FEAP seit den 80-er Jahren des 20-en Jahrhunderts entwickelt wird [8], ist die Grundstruktur in FORTAN 77 implementiert. Die Sprache bietet daher nur statische Platzverwaltung. Dennoch haben die Entwickler die FE-Matrizen durch die Verwendung von Pointern[16] dynamisch alloziert. Die FE-Werte werden in der History-Variable `hr(np(...))` festgehalten. Wobei `np(...)` der Pointer ist. Alle Informationen zu den Pointern sind in der Datei `palloc.f` festgelegt. Die wichtigsten Werte sind:

```
...
  data    (nlist(i),i=1,list)/
...
!     Mesh arrays
&         'D', 'DR', 'F', 'F0', 'FPRO', 'FTN',
&         'ID', 'IE', 'IX', 'LD', 'P', 'S',
&         'SLODI', 'T', 'TL', 'U', 'UL', 'VEL',
&         'X', 'XL', 'ANG', 'ANGL',
! 'D    ',      !    25: Material parameters
! 'DR   ',      !    26: Residual/reactions
```

[14] Die sind auf FEAP Webseite zu finden: `http://www.ce.berkeley.edu/projects/feap/`
[15] in FEAP *user element*
[16] Pointer oder Zeiger (*deutsch*) ist eine Variable, die die Adresse einer Variablen enthält [36]

```
! 'F    ',     !    27: Nodal load/displacement, current
! 'F0   ',     !    28: Nodal load/displace pattern, base
! 'FPRO ',     !    29: DOF proportional load numbers
! 'FTN  ',     !    30: Nodal load/displacement, current
! 'ID   ',     !    31: Equation numbers/boundary conds
! 'IE   ',     !    32: Element assembly information
! 'IX   ',     !    33: Element connection data
! 'LD   ',     !    34: Element local/global eq numbers
! 'P    ',     !    35: Element vector
! 'S    ',     !    36: Element array
! 'SLODI',     !    37: Surface load data
! 'T    ',     !    38: Nodal temperatures
! 'TL   ',     !    39: Nodal temperaturese, element
! 'U    ',     !    40: Nodal solutions/increments
! 'UL   ',     !    41: Nodal solutions/increments,element
! 'VEL'  ,     !    42: Nodal transient solution values
! 'X    ',     !    43: Nodal coordinates
! 'XL   ',     !    44: Nodal coordinates, element
! 'ANG  ',     !    45: Nodal angles
! 'ANGL ',     !    46: Nodal angles, element
...
```

Code 2.2: Wichtige Pointer in FEAP.

FEAP benutzt für die Knotenkräfte und die Residuen den gleichen Pointer. Daher muss im `BATCh`-Modus der Befehl `reac` aktiv sein, um die Reaktionskräfte zu erhalten.

Kapitel 3.

Entwicklung des Optimierungsalgorithmus

Nachdem die Grundlagen im Kapitel 2 erarbeitet sind, kann jetzt auf die Entwicklung der tatsächlichen Algorithmen eingegangen werden. Wie bereits im Kapitel 1 erwähnt, werden drei Algorithmen vorgestellt, nachfolgend validiert und im Anschluss diskutiert.

Als Berechnungsmodell wurde eine Kragscheibe, wie in der Abbildung 3.1 dargestellt, gewählt, die linear elastisch gerechnet wird. Der Grund für die Wahl eines solchen Modells ist die Existenz einer analytischen Lösung, so dass man auf die Experimente verzichten kann, was die Validierung enorm erleichtert.

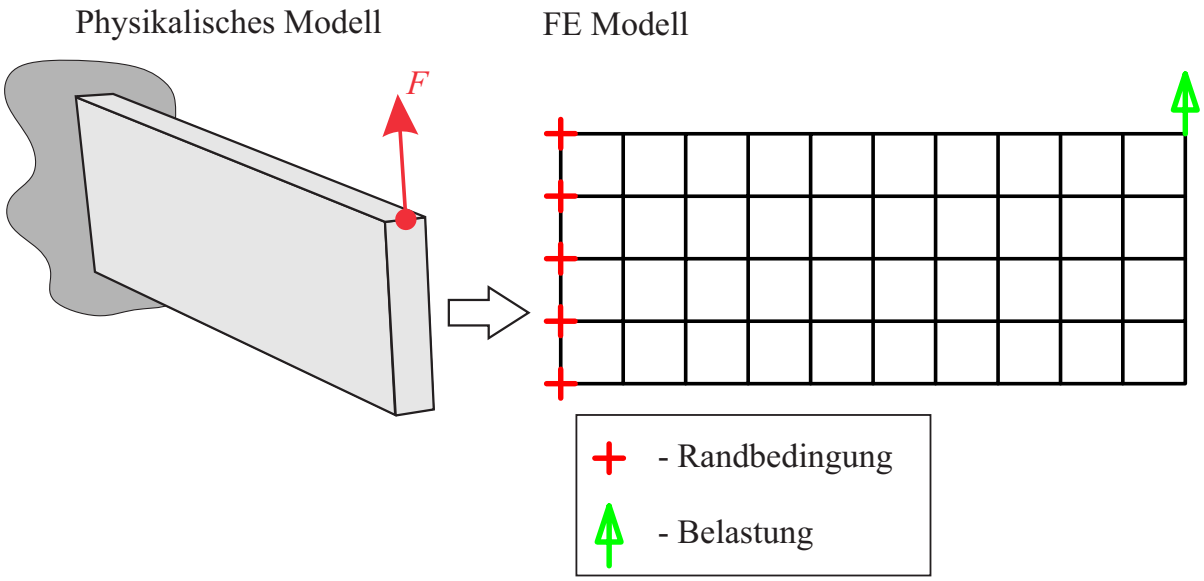


Abbildung 3.1.: Modellierung.

Wie man dem obigen Bild entnehmen kann, wird das physikalische Modell diskretisiert und mit den Randbedingungen und Belastungen versehen. Auf diese Weise entsteht das FE Modell. Zusätzlich wird das Materialgesetz formuliert und letztendlich der Solver[1] passend eingestellt. Weitere Details zur Modellierung werden im Abschnitt 3.2 zur Validierung näher erläutert.

Für so ein elementares Modell ist es möglich eine geschlossene-analytische Lösung zu

[1]Solver oder Gleichungslöser übernimmt die eigentliche Berechnung: Assemblierung der Gesamtmatrix und anschließende Lösung des Gleichungssystems [35]

erzeugen. Allerdings ist es eine Ausnahme und im Allgemeinen hat man komplexere Systeme, so dass man nur numerisch oder experimentell die Werte ermitteln kann. Dennoch muss die analytische Lösung an die experimentelle Form angepasst werden, um die Algorithmen vollwertig validieren zu können. Um das erreichen zu können, wird die Lösung in diskrete Abschnitte unterteilt. Ab jetzt wird die analytische unterteilte Lösung als die quasi-experimentelle Messung betrachtet. Die Abbildung 3.2 stellt die Idee der diskreten Aufteilung schematisch dar.

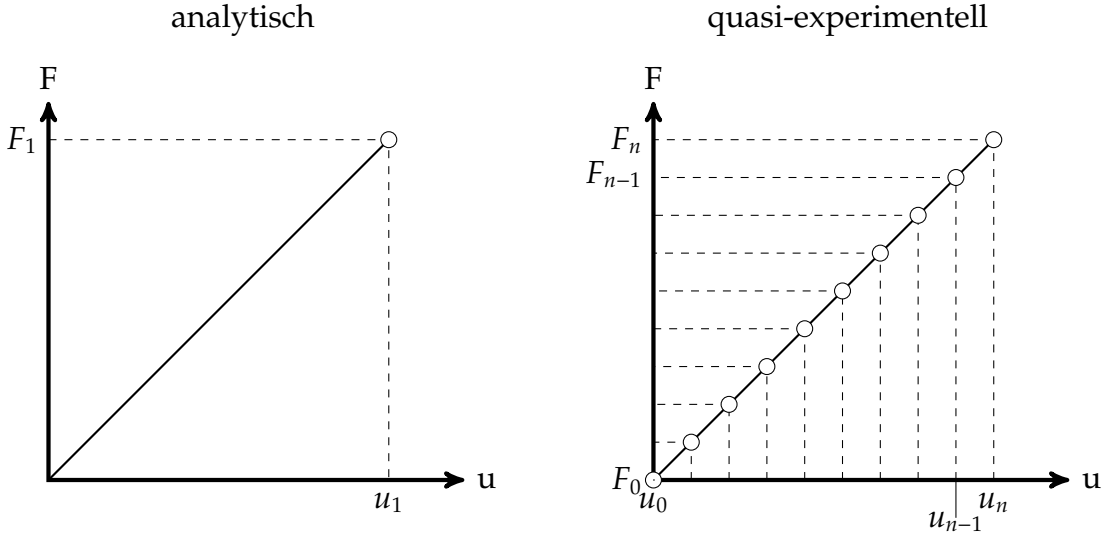


Abbildung 3.2.: Analytische und quasi-experimentelle Darstellung.

Der Index n steht dabei für die Anzahl der Messwerte.

Bei der Entwicklung der Algorithmen wurden unter anderem die Zufallszahlen benutzt. Allerdings ist der Rechner zwar eine sehr genaue, aber auch eine deterministische Maschine. Was zur Folge hat, dass man streng genommen keine Zufallszahlen, sondern die Pseudo-Zufallszahlen mit Hilfe von Algorithmen generieren kann [48]. Die Erzeugung in Fortran[2] 90 erfolgt über interne Subroutinen `random seed` und `random number`. Auf die Details zur Implementierung wird im Abschnitt 3.1 eingegangen.

3.1. Implementierung

In diesem Abschnitt wird auf die Implementierung eingegangen. Dabei werden die drei Algorithmen dem Leser näher erläutert. Zuerst soll die Grundstruktur der Modellierung mit FEAP dargestellt werden.

Programmstruktur

Die Kommunikation zwischen dem Benutzer und dem Programm erfolgt, anders als bei der meisten kommerziellen FE-Software, über die Textdateien. Möchte man die

[2] Eine standartisierte Methode zur Generation der Zufallszahlen, bzw. der Pseudo-Zufallszahlen existiert erst ab Fortran 90 [29]

Berechnung mit dem gleichen Modell, aber eventuell anderen Materialparametern mehrmals durchführen, so gibt es in FEAP eine Möglichkeit den File mit den Befehlen in zwei Dateien aufzuteilen. Die Abbildung 3.3 illustriert das Prinzip. Die `Imain`-Datei beinhaltet die Initialisierung der gesuchten Parameter, die Zuweisung auf die Datei mit den Befehlen für Modellaufbau, die Obergrenze für die Schleife und die Befehle für den eigentlichen Algorithmus. Die `Ifile`-Datei enthält alle Befehle zur Modellierung, folglich Informationen zum Werkstoff, Elementtyp, Randbedingungen, Belastung, etc.

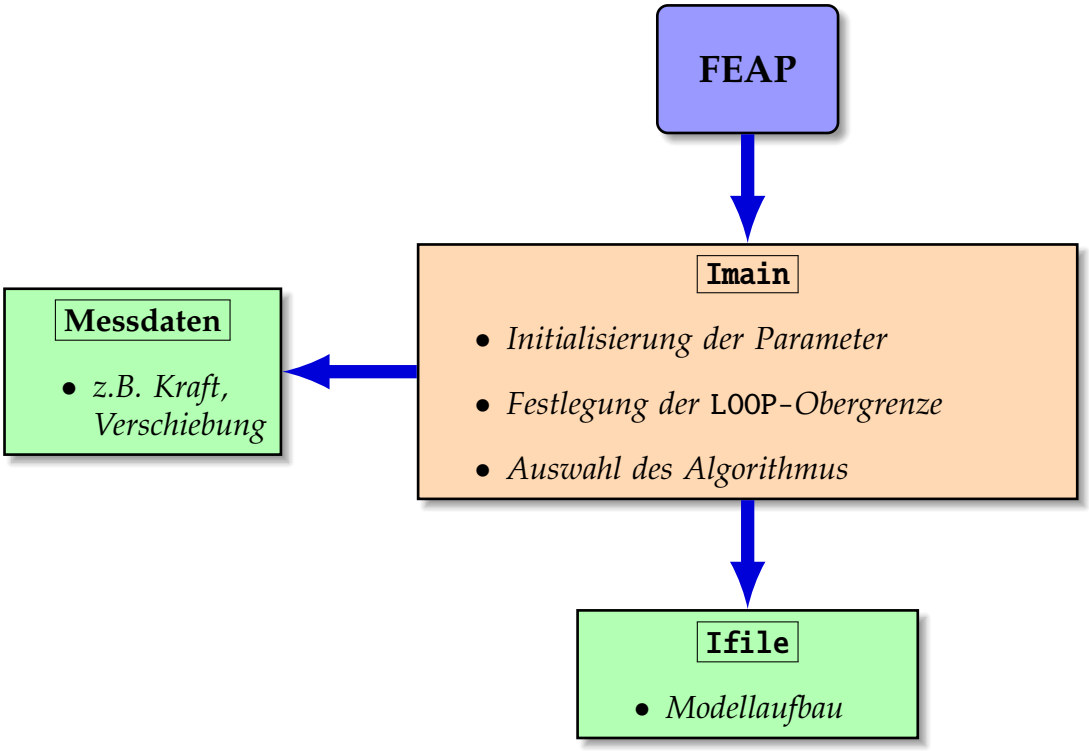


Abbildung 3.3.: Filesystem während der Berechnung.

Die `Imain`-Datei und die `Ifile`-Datei sind in dem Anhang A.1, A.2 zu finden.

Zugleich greift die `Imain`-Datei mittels des `POPT`-Befehls (vgl. Code 3.1) auf die Messdaten zu. Die Daten werden eingelesen und in einem Array gespeichert. Der Code 3.1 zeigt die nötige Eingabe der Steuerparameter. Die `Obergrenze` stellt die maximale Anzahl der Durchläufe dar. Konvergiert der Algorithmus nicht, so wird nach dem Erreichen der `Obergrenze` die Berechnung abgebrochen. In FEAP ist es möglich maximal 999 Berechnungen durchzuführen. Der Befehl `INCLude Ischeibe` referenziert auf das eigentliche Modell (vgl. Abbildung 3.3).

```
LOOP,Obergrenze
  INCLude Ischeibe
  POPT,Algo,Belastung,mein_Parameter,Richtung+Knoten,Messdaten
NEXT
```

Code 3.1: `LOOP`-Konstrukt in FEAP.

Letztendlich beinhaltet der `POPT` den Kern der Optimierung. Die Tabelle 3.1 fasst die Einträge zusammen:

Eintrag	Beschreibung
`Algo`	die Auswahl des Algorithmus. Es dürfen Namen mit vier Zeichen genutzt werden
`Belastung`	man soll die Belastungsart angeben. `force` steht für die Kraftbelastung und `disp` für die Verschiebungsrandbedingung
`mein_Parameter`	gesuchte Parameter
`Richtung + Knoten`	man soll die Richtung der Belastung angeben und den Knoten, der die gleiche Position wie die Messstelle haben
`Messdaten`	der Name der Datei mit den Messdaten. Die Datei soll im gleichen Ordner wie die FEAP-Datei sein.

Tabelle 3.1.: Steuerparameter der Algorithmen.

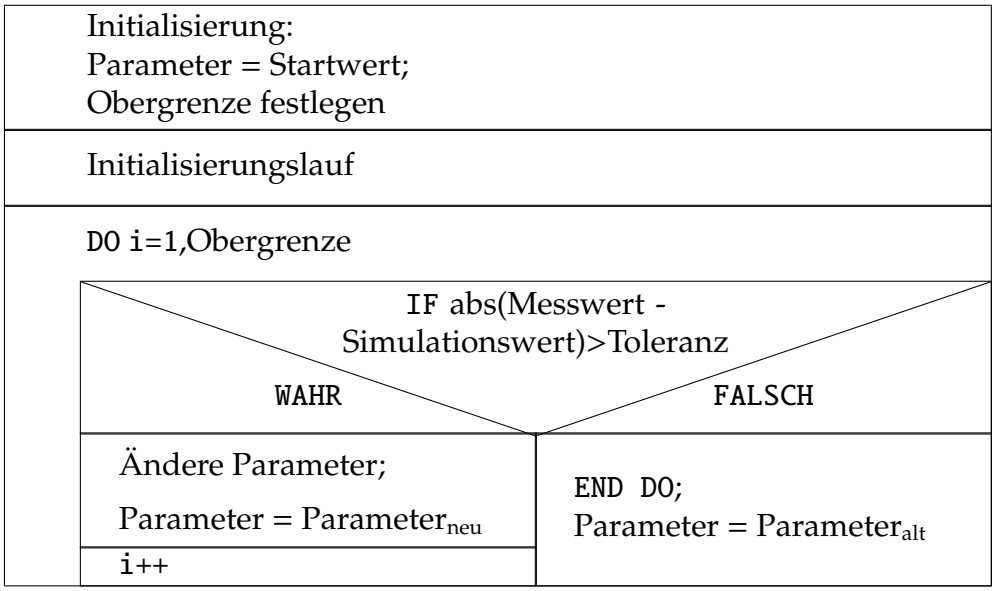


Abbildung 3.4.: Ablauf der Berechnung.

Allgemein wird die Berechnung, unabhängig von dem gewählten Algorithmus, nach einem Schema durchgeführt. Die Abbildung 3.4 illustriert den Ablauf.

Im ersten Schritt wird die Obergrenze für die `LOOP`-Schleife festgelegt. Hier sollte man die Zahl nicht zu klein wählen, ansonsten wird der Algorithmus während der Konvergenz abgebrochen. Zur Initialisierung wird der Wert des Parameters geschätzt und im nächsten Schritt ein Initialisierungslauf durchgeführt. Auf diese Weise sind die FEAP-interne Variablen besetzt und man kann die `LOOP`-Schleife starten, die nichts anderes als die bekannte `DO`-Schleife ist. In der Schleife wird bei jedem Schritt die Bedingung geprüft, ob der die Differenz zwischen dem Mess- und Simulationswert größer als die vorgegebene Toleranz ist. Ist die Abzweigung falsch, so hat man das Optimum erreicht, wenn die Bedingung wahr ist, so geht man zu der nächsten Iteration.

Als Benutzer sieht man nur die Änderung der numerischen Werte des Parameters und

die Deformation des FE-Modells. So kann man optisch nicht erkennen, ob der Algorithmus konvergiert oder divergiert, was nicht praktisch ist. Aus diesem Grund wurde zusätzlich das Plotten mit dem Gnuplot[3] implementiert. Die Freischaltung erfolgt in der LOOP-Schleife mit gnup-Befehl. Der Code 3.2 zeigt die Benutzung des Befehls:

```
LOOP,Obergrenze
  INCLude Ischeibe
  POPT,Algo,Belastung,mein_Parameter,Richtung+Knoten,Messdaten
  gnup,Simulationsdaten,Messdaten
NEXT
```

Code 3.2: Gnuplot in FEAP main-Datei.

In jeder Iteration werden im gleichen Plot die Mess- und Simulationsdaten dargestellt. Die Voraussetzung ist, dass der Benutzer Gnuplot auf seinem Rechner installiert hat und die PATH-Variable eingetragen hat. Die komplette Subroutine ist in dem Anhang A.5 zu finden.

Nach jeder Iteration werden alle nötigen Werte in die FEAP interne Output-Datei geschrieben und nach dem Erreichen des Optimums wird eine externe Datei mit der Zusammenfassung der Berechnung erstellt. Aus der Datei können die optimale, FE- und Fehlerwerte rausgeholt werden.

Zufallszahlen

Zur Bestimmung der Schrittweite werden in Algorithmen RS und RSR die Zufallszahlen benutzt. Wie am Anfang des Kapitels 3 erwähnt werden bei den deterministischen Rechnern keine echten Zufallszahlen benutzt, sondern die Pseudo-Zufallszahlen. Die allerdings nicht intrinsisch in FORTRAN 77 bzw. FEAP enthalten sind. Demnach wird eine Subroutine, die im Code 3.3 zu sehen ist, zur deren Erzeugung implementiert.

```
SUBROUTINE randArray(array,n)
  IMPLICIT NONE
  real*8           :: randnr
  integer          :: m,i,n
  integer,dimension(8) :: date_time
  integer,dimension(1) :: seed
  real*8,dimension(n)  :: array

  call date_and_time(values=date_time)

  call random_seed
  call random_number(randnr)

  call random_seed(size=m)

  seed(1) = date_time(6)*date_time(7)+date_time(8)

  call random_seed(put=seed)
```

[3]Software zur Visualisierung von Funktionen und Daten, aus http://www.gnuplot.info

```
  do i = 1,n
    call random_number(randnr)
    array(i) = randnr
  end do

END SUBROUTINE
```

Code 3.3: Subroutine zur Generierung der Pseudo-Zufahlszahlen.

Dabei wird ein eindimensionales Array mit `n` Zahlen erzeugt, im folgenden Bereich gültig sind: $n_i \in [0, 1[,\ \forall i \in \mathbb{N}^*$.

Randbedingungen

Bei der Simulation muss man darauf achten, dass alle Randbedingungen exakt mit denen der Messung übereinstimmen. So trivial das auch klingt, begeht man bei der Nichterfüllung dieser Voraussetzung einen großen Fehler und erreicht keine plausiblen Ergebnisse. Belastet man die Probe im Versuch und das Modell in der Simulation mit einer Kraft, so verlaufen die Mess- und Simulationskurven entlang der belasteten Kraft. Die Abbildung 3.5 illustriert das Prinzip.

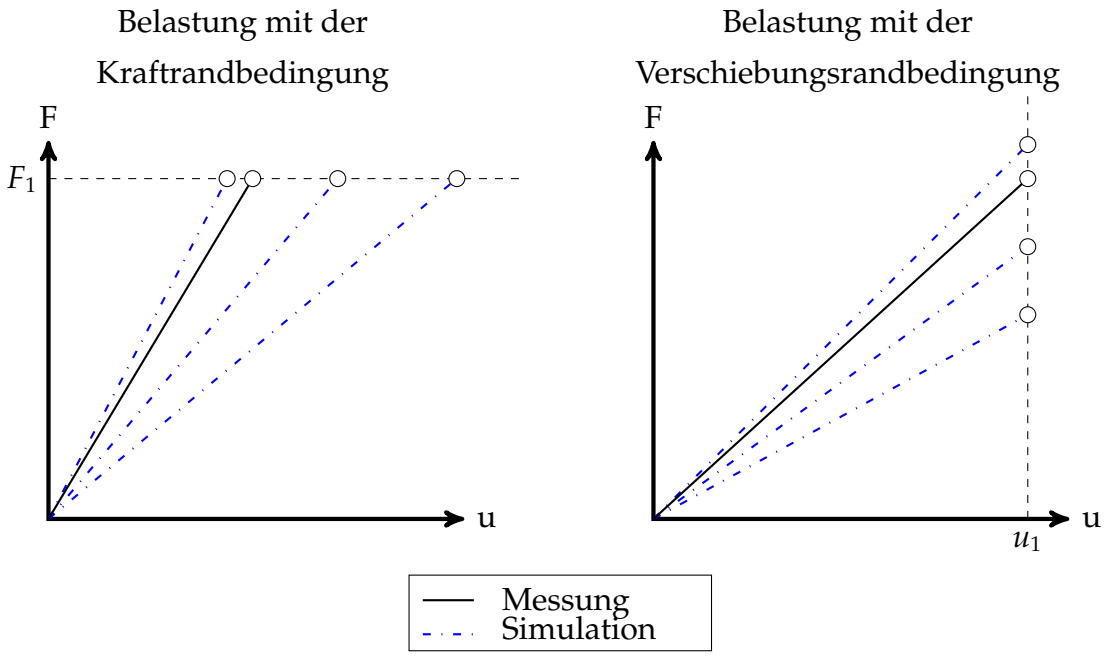


Abbildung 3.5.: Unterschied zwischen Kraftbelastung und der Verschiebungsrandbedingung.

Die schwarze Linie ist der Messwert und die blauen sollen die Simulationskurven mit unterschiedlichen Parametern darstellen. Wie man sieht, übersteigen die Linien nicht die belastete Kraft. Hat man anstelle der Kraft eine Verschiebungsrandbedingug, so ist das Verhalten analog. Diese Eigenschaft des maximalen Wertes erleichtert die Berechnung mit der Methode der kleinsten Quadrate. Und zwar bestimmt man die Differenz zwischen dem Experiment und der Simulation nur in Bezug auf eine Achse.

Alorithmennamen

Es wurden drei Algorithmen zur Lösung des Problems entwickelt. Um die besser zuordnen zu können werden sie mit folgenden Bezeichnungen eingeführt:

- Algorithmus 1: Random Step oder RS Algorithmus,
- Algorithmus 2: Differenzalgorithmus oder Diff Algorithmus,
- Algorithmus 3: Random Step Residuen Algorithmus oder RSR Algorithmus.

3.1.1. RS Algorithmus

Bei dem Algorithmus handelt es ich, um ein zufallsbasiertes Verfahren. Dabei wird ein Punkt zur Referenz gezogen. Die Abbildung 3.6 verdeutlicht das Prinzip anhand des Kraft-Verschiebungsdiagramms.

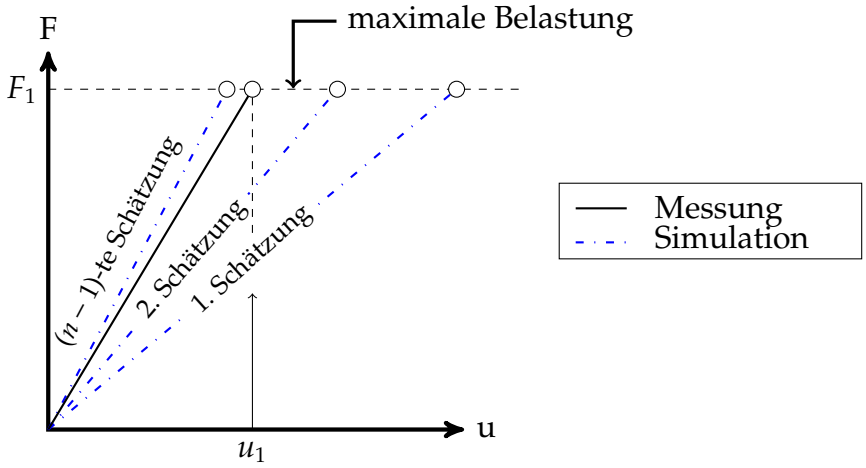


Abbildung 3.6.: Schematischer Ablauf des RS Algorithmus.

Bei der ersten Schätzung des Parameters wird auch gleichzeitig eine Zufallszahl, die mit dem Pseudo-Zufallsgenerator (siehe Code 3.3) erzeugt war, generiert, die im nächsten Schritt mit dem Parameter multipliziert wird. Die Schrittweite ist variabel gestaltet, so dass man bei großer Differenz zwischen gemessenen und simulierten Werten der Schritt grob ist und bei Annäherung zum Optimum die Schritte kleiner werden. Auf diese Weise wird die Konvergenz erzielt. Da der Algorithmus mit dem Zufallsgenerator arbeitet, kann man nicht vorhersagen wie viele Iterationen bis zur Konvergenz nötig sind.

Die Abbildung 3.7 verdeutlicht den Ablauf des RS Algorithmus.

Subroutine `algo_random(ParameterName,val,val2,val_umess,val_usim, Toleranz, Knoten, Dateiname, reac,disp, measMatrix, nrrows, nrcols, SimVekF, SimVekU)`: Identifizierung des Parameters `ParameterName`

Variablen:

`ParameterName`	{`character`-Variable}
`val1, val2`	{`real`-Variablen}
`val_mess, val_sim`	{`real`-Variablen}
`korpar`	{`real`-Variable}
`Toleranz`	{`real`-Variablen}
`reac,disp`	{`real`-Variablen}
`measMatrix`	{`array(:)(:)`-Variable}
`nrrows, nrcols`	{`integer`-Variablen}
`SimVekF,SimVekU`	{`array(:)`-Variablen}
`error,prt`	{`logical`-Variablen}

Initialisierung:
`call getparam(ParameterName, val, error, prt)` = man holt den Wert des Parameters aus System und weist den der Variable `val` zu;
`call randArray(randnrs,n)` = Zufallszahlen werden

`if abs(umess-usim) >Toleranz`

WAHR

FALSCH: `call plstop()` = Berechnung wird gestoppt. Output Dateien werden geschrieben.

`a:=abs(umess-usim)>Toleranz`

$a \times 1e+03$	$a \times 1e+02$	$a \times 1e+01$	a
`korpar` = `randnrs`	`korpar` = $\frac{\texttt{randnrs}}{100}$	`korpar` = $\left(\frac{\texttt{randnrs}}{100}\right)^2$	`korpar` = $\left(\frac{\texttt{randnrs}}{100}\right)^3$

`if (umess < usim)`

WAHR: `val2 += val × korpar`

FALSCH: `val2 -= val × korpar`

`val2 :=` neuer Parameter

∅

Abbildung 3.7.: Verlauf des RS Algorithmus.

Der vollständige Code zum RS Algorithmus ist im Anhang A.6 zu finden.

3.1.2. Diff Algorithmus

Dieser Algorithmus basiert ebenfalls wie der RS Algorithmus auf der Auswertung eines Punktes. Dabei verzichtet man auf den Zufallsgenerator und bestimmt die Differenz zwischen der Simulation und dem Experiment. Auf diese Weise braucht man um besten Fall nur eine Schätzung, um den gesuchten Parameter zu identifizieren. Die Abbildung 3.8 illustriert das Vorgehen.

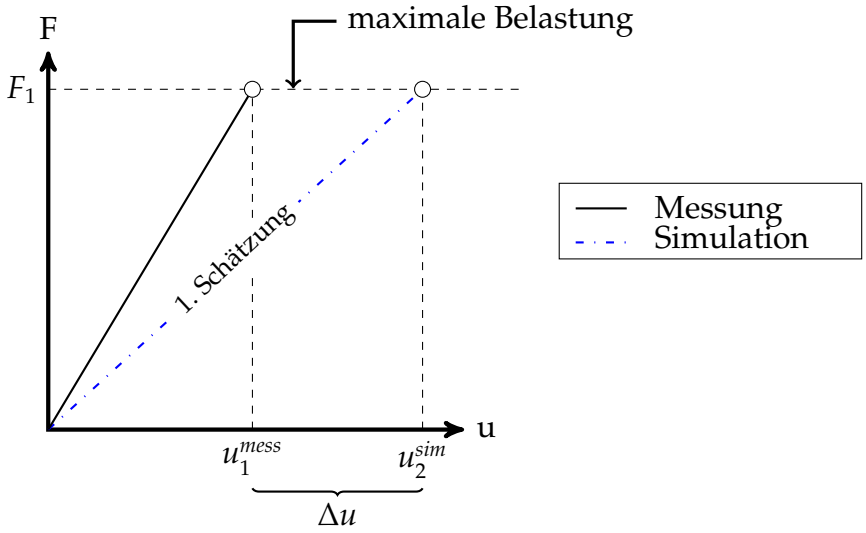


Abbildung 3.8.: Schematischer Ablauf des Diff Algorithmus.

Die Differenz Δu wird gebraucht, um den Faktor f auf folgende Art zu bestimmen:

$$\Delta u = u_1^{mess} - u_2^{sim}, \qquad f = \frac{\Delta u}{u_1^{mess}}. \tag{3.1}$$

Der Faktor f als Multiplikator für den alten Parameter benutzt. Auf ähnliche Weise wurde im RS Algorithmus die Pseudo-Zufallszahl benutzt.

Die genauere Struktur des Diff Algorithmus ist der Abbildung 3.9 zu entnehmen.

Subroutine `algo_diff(ParameterName,val,val2,val_umess,val_usim, Toleranz, Knoten, Dateiname, reac,disp, measMatrix, nrrows, nrcols, SimVekF, SimVekU)`: Identifizierung des Parameters `ParameterName`

Variablen:

`ParameterName`	{`character`-Variable}
`val1, val2`	{`real`-Variablen}
`val_mess, val_sim`	{`real`-Variablen}
`faktor`	{`real`-Variable}
`Toleranz`	{`real`-Variablen}
`reac,disp`	{`real`-Variablen}
`measMatrix`	{`array(:)(:)`-Variable}
`nrrows, nrcols`	{`integer`-Variablen}
`SimVekF,SimVekU`	{`array(:)`-Variablen}
`error,prt`	{`logical`-Variablen}

Initialisierung:
`call getparam(ParameterName, val, error, prt)` = man holt den Wert des Parameters aus System und weist den der Variable `val` zu

`if abs(val_mess-val_sim) >Toleranz`

WAHR

`faktor = (val_sim-val_mess)/val_mess`

`if (val_mess < val_sim)`

WAHR

`val2 += val × faktor`

FALSCH

`val2 -= val × faktor`

`val2 :=` neuer Parameter

FALSCH

`call plstop()` = Berechnung wird gestoppt. Output Dateien werden geschrieben.

∅

Abbildung 3.9.: Verlauf des Diff Algorithmus.

Der komplette Fortran Code zu dem Algorithmus ist im Anhang A.7 hinterlegt.

3.1.3. RSR Algorithmus

Der letzte Algorithmus basiert auf der Methode der kleinsten Quadrate. In jedem Iterationsschritt werden die Residuen bestimmt und man entscheidet wie der nächste Schritt sein wird, anhand der Summe der Quadrate der Residuen. Die Abbildung 3.10 illustriert das Prinzip.

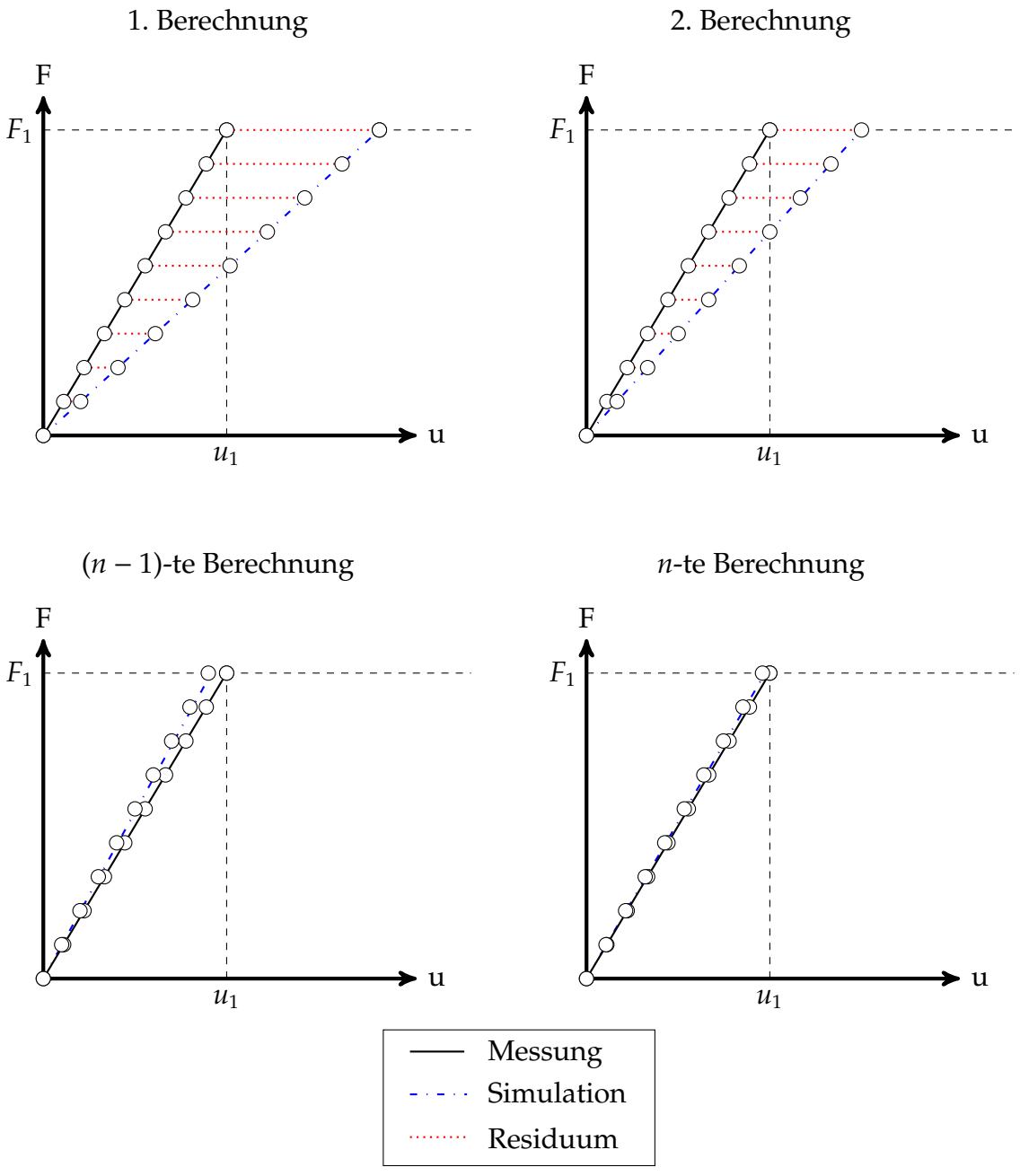


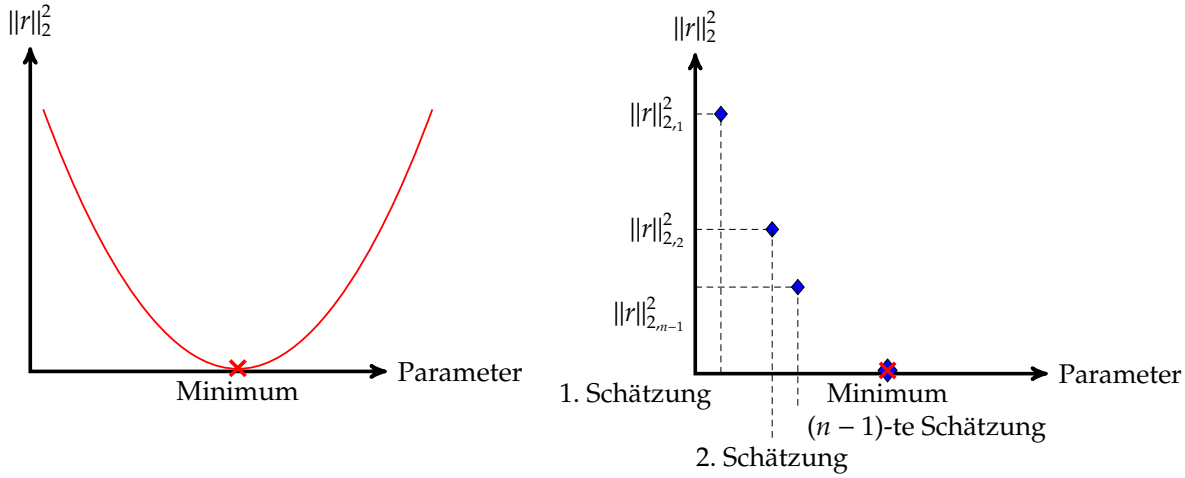
Abbildung 3.10.: Schematischer Ablauf des RSR Algorithmus.

Bei der Annäherung der Mess- und Simulationskurven konvergiert $\|r\|_2^2$ gegen ein Bestimmtes Minimum und zwar gegen die Null. Denn erst wenn die Summe der kleinsten Quadrate bei den Kurve verschwindet, kann man davon ausgehen, dass

die Simulation dem Experiment entspricht und die gesuchten Parameter richtig identifiziert sind. Denn die Simulation in ihrer Aufbau das Experiment imitiert. Da die numerischen Berechnungen gewisse Fehler aufgrund der deterministischen Eigenschaft des Rechners aufweisen, hat man eine Toleranz eingefügt, um die Simulationen nicht in eine Endlosschleife zu versetzen.

Als nächster wichtiger Punkte sollte das Speichermanagement angesprochen werden. Im Abschnitt 3.1 unter Programmstruktur wurde schon diskutiert, dass die mehrfache Ausführung des Modells mittels des `LOOP`-Konstrukts erfolgt, das der konventionellen `DO`-Schleife entspricht. Allerdings mit einer Ausnahme. Nämlich werden nach jeder Iteration alle FEAP interne Variablen aus dem Programmspeicher gelöscht, so dass man nicht auf die frühere Berechnungsergebnisse direkt zugreifen kann. Für die Implementierung der Methode der kleinsten Quadrate heißt es, dass man nach der Berechnung keine Informationen über die frühere Werte der Summe der Quadrate mehr hat. Um dennoch mindestens zwei Residuen vergleichen zu können, wurde der Informationsaustausch über externe Dateien realisiert.

Im Gegensatz zur Optimierung der analytischen Funktion, hat man bei der inversen Parameteridentifikation mit FEAP lediglich diskrete Werte. Die Abbildung 3.11 illustriert das Prinzip.



(a) Analytische Werte.

(b) Diskrete Werte.

Abbildung 3.11.: Analytischer Kurve und diskreten Werten von FEAP.

Wie man sieht, sind im Diagramm 3.11 (b) nur die Werte der einzelnen Simulationen bekannt. Wie der Verlauf zwischen den Punkten ist, bleibt unbekannt. Man kann entweder zwischen den Punkten zu interpolieren, was gewisse Ungenauigkeit mit sich bringt, oder mehrere Simulationen durchführen, was nicht die effizienteste Vorgehensweise ist. Bei der Implementierung des RSR Algorithmus wurde keine der beiden Möglichkeiten gewählt, sondern man ging von einer bis zur nächsten Simulation, solange die Summen der Residuen sanken. In der Abbildung 3.12 ist die Struktur des Algorithmus dargestellt.

Subroutine `algo_rnd_stps(ParameterName, val, val2, val_mess, val_sim,Toleranz,Knoten, Dateiname, reac, disp, measMatrix, nrrows, nrcols, SimVekF, SimVekU)`: Identifizierung des Parameters `ParameterName`

Variablen:

`ParameterName`	{`character`-Variable}
`val1, val2`	{`real`-Variablen}
`val_mess, val_sim`	{`real`-Variablen}
`faktor`	{`real`-Variable}
`Toleranz`	{`real`-Variablen}
`reac,disp`	{`real`-Variablen}
`measMatrix`	{`array(:)(:)`-Variable}
`nrrows, nrcols`	{`integer`-Variablen}
`SimVekF,SimVekU`	{`array(:)`-Variablen}
`error,prt`	{`logical`-Variablen}
`SSR2`	{`real`-Variable}
`res1,res2`	{`real`-Variablen}

Initialisierung:
`call getparam(ParameterName, val, error, prt)` = man holt den Wert des Parameters aus System und weist den der Variable `val` zu ; `inquire(file="ssr.txt",exist=vorhanden)` = es wird geprüft, ob Datei mit Residuen vorhanden ist; `call randArray(randnrs,n)` = Pseudo-Zufallszahlen werden ezeugt; `SSR2 = SumResQuadr(val_mess,val_sim)` = Summe der Quadrate der Residuen wird gebildet

`if (vorhanden)`

WAHR

Residuen werden ausgewertet.

`if (res1 > res2)`

WAHR

open(unit = 999, file= "ssr.txt",position="append") write(999,*) SSR2

FALSCH

`val2 -= val ×` `randnrs(1)/2.0`

FALSCH

`val2 = val + val`

∅

Daten für Gnuplot werden geschrieben

`SSR2 > Toleranz`

WAHR

Schreib SSR2 in "SSR.txt"

FALSCH

`call plstop()`

Abbildung 3.12.: Verlauf des RSR Algorithmus.

Steigt der Wert der Summe der Residuen, so wird das vom Algorithmus abgefangen und der Iterationsschritt ändert die Richtung. Auf diese Weise wird versucht das Überspringen des Minimums zu vermeiden.

3.2. Validierung

Nach dem alle Grundlagen erläutert und die drei Algorithmen vorgestellt sind, sollen die Ergebnisse der Numerik validiert werden.

Das im Bild 3.1 gezeigte Modell wird der Validierung dienen. Die geometrischen Abmessungen der Kragscheibe sind in der Abbildung 3.13 skizziert.

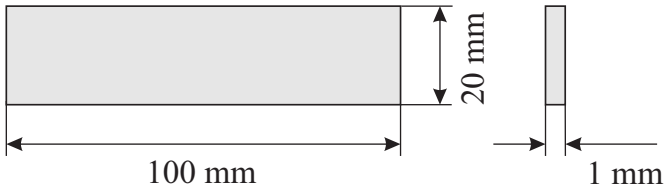


Abbildung 3.13.: Abmessungen der Kragscheibe.

Als Materialeigenschaft braucht man lediglich das Elastizitätmodul $E = 2.1e+05\ MPa$, weil man beim Vergleich der Daten nur die Durchbiegung bei gegebener Kraftbelastung betrachtet.

Analytisch ergibt sich die Durchbiegung nach der Balkentheorie mit der Differentialgleichung:

$$(EIw'')'' = q. \tag{3.2}$$

Mit gegeben Randbedingungen und Last lässt sich die Gleichung 3.2 folgendermaßen umformen:

$$w(x) = \frac{F\,l^3}{6EI}\left(-\frac{x^3}{l^3} + 3\frac{x^2}{l^2}\right), \tag{3.3}$$

bzw. die maximale Durchbiegung ist dann:

$$w(x = l) = w_{\max} = \frac{F\,l^3}{3EI}. \tag{3.4}$$

Mit den Abmessungen aus Abbildung 3.13 und dem Elastizitätsmodul $E = 2,1\text{e} + 05\ MPa$ bekommt man für die maximale Durchbiegung infolge der Kraft $F = 1000\ N$ einen numerischen Wert von:

$$w_{max} = 2,38\ mm.$$

FEAP liefert zum Vergleich mit gegebenen Daten einen Wert von `2,3809e+00` mm. Dabei wurde linear elastisch mit `QUAD4`-Elementen und dem ebenen Spannungszustand gerechnet. Die Tabelle 3.2 fasst alle relevanten Information zusammen:

	F $[N]$	E $[MPa]$	w_{max} $[mm]$
analytisch	1000	2,1e+05	2,380952
FEAP	1000	2,1e+05	2,38092

Tabelle 3.2.: Relevante Werte zur Berechnung.

Die Anzahl der Nachkommastellen ist zwar physikalisch nicht brauchbar, aber um die Algorithmen besser numerisch Vergleichen zu können, nimmt man die 6 Nachkommastellen mit. Somit sind alle nötigen Information zur Validierung gesammelt. Das Ziel ist dabei mit allen drei Algorithmen den Wert des Elastizitätsmodul zu bestimmen. Als Ausgangswert startet man mit einem willkürlich gewählten Elastizitätsmoduls $E =$ 0,9e+05 MPa.

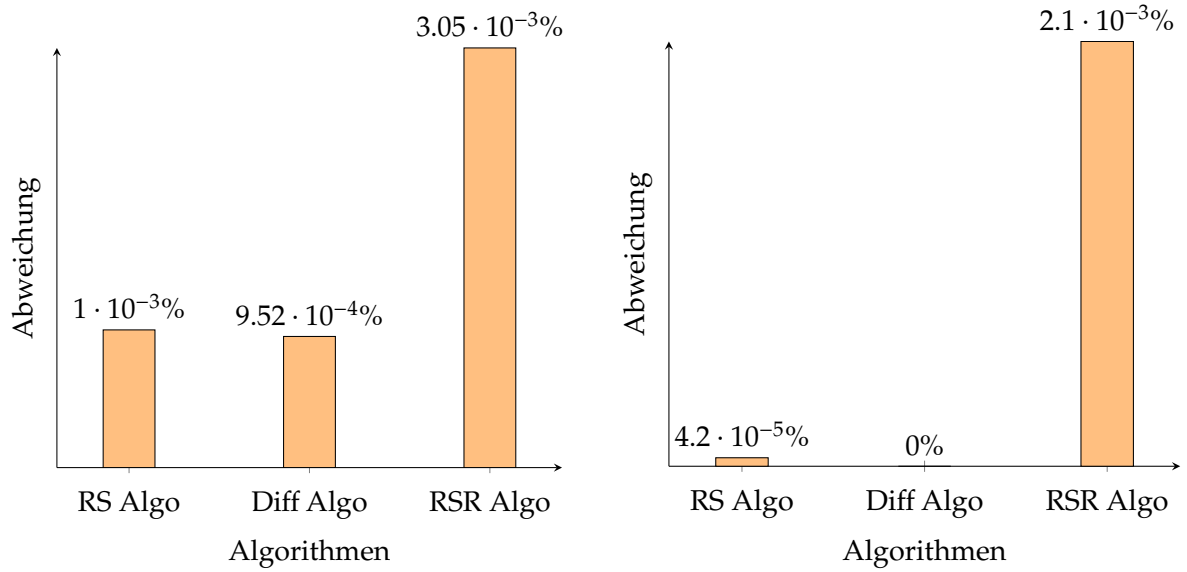
Man beachte, dass die Algorithmen RS und RSR zufallsbasiert sind und die Anzahl der Iterationen auch mit gleichen Werten abweichen kann. Ebenfalls ist der Startwert bei den beiden Algorithmen äußerst wichtig. Wird der Start weit vom Optimum gewählt, so braucht das Programm mehr Iterationen, um Minimum zu finden. So hat der RSR Algorithmus bei einem Startwert $E = 1000\,MPa$ genau 319 Iterationsschritte gebraucht. Der relative Fehler betrug dabei 0,078 %, was deutlich im akzetablen Bereich liegt. Die Tabelle 3.3 fasst die Ergebnisse der Berechnungen für $E =$ 0,9e+05 MPa zusammen:

	w_{max} $[mm]$	E $[MPa]$	Anzahl Iterationen	$\|\|r\|\|_2^2$
RS Algorithmus	2,380953	2,099979e+05	39	0,000000
Diff Algorithmus	2,380952	2,099980e+05	2	0,000000
RSR Algorithmus	2,381002	2,099936E+05	15	0,000000

Tabelle 3.3.: Vergleich der Algorithmen.

Dagegen braucht Diff Algorithmus zwei Iterationen, weil bei dem Verfahren die Differenz ausgerechnet wird. Die Anzahl der Iterationen bei Algorithmen RS und RSR ist der gerundete Mittelwert aus drei Durchläufen.

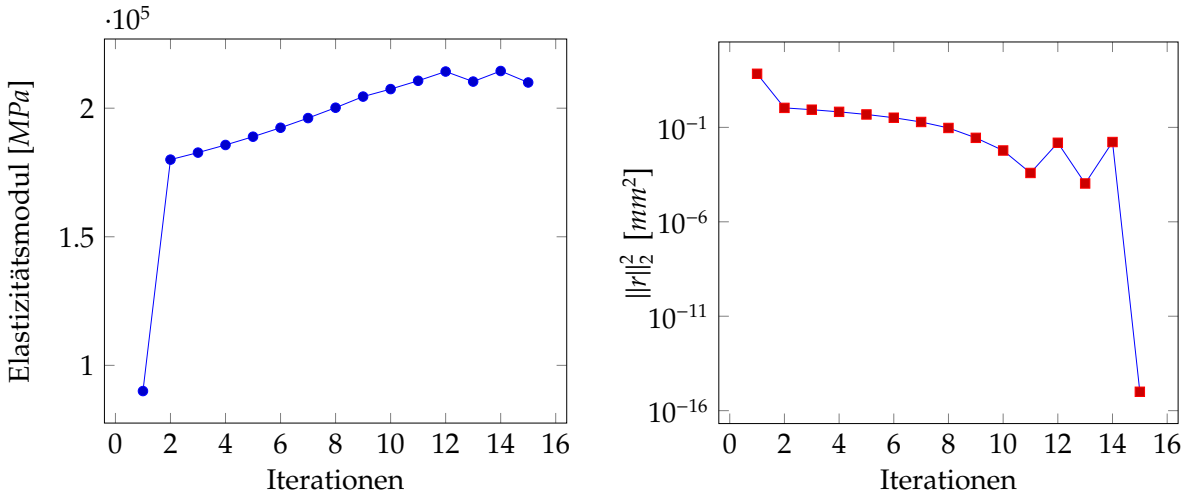
Laut der Tabelle 3.3 ist die Summe der Quadrate der Residuen Null. Was lediglich ein Darstellungsproblem ist, weil man nur 6 Nachkommastellen sieht. Die Summe ist so klein, dass man mehr Nachkommastellen mitnehmen müsste. Numerische Werte sind sehr mühsam zu vergleichen, darum wird in der Abbildung 3.14 der relative Fehler betrachtet. Betrachtet man das gesuchte Elastizitätsmodul (vgl. Abbildung 3.14 (a)) oder die maximale Durchbiegung(vgl. Abbildung 3.14 (b)), so liegt in beiden Fällen der Fehler bei allen drei Algorithmen deutlich unter 1 %.



(a) Abweichung des Elastizitätsmoduls. (b) Abweichung der max. Durchbiegung.

Abbildung 3.14.: Relativer Fehler der Algorithmen.

Die Tabelle 3.3 und die Abbildung 3.14 beziehen sich auf den letzten Iterationsschritt mit den finalen Werten. Um den Verlauf des gesuchten Parameters und der Summe der Quadrate der Residuen zu verfolgen, werden die Werte über alle Iterationsschritte betrachtet. In der Abbildung 3.15 werden die Verläufe am Beispiel des RSR Algorithmus veranschaulicht:



(a) Verlauf des gesuchten Parameters, E. (b) Verlauf der Quadrate der Residuen.

Abbildung 3.15.: Änderung der Werte über Iterationen anhand des RSR Algorithmus.

Zusammenfassend lässt sich sagen, dass alle drei Algorithmen mit ausreichender Genauigkeit, wobei der größte relativer Fehler im Promillebereich liegt, die inverse Parameteridentifikation geschafft haben. Über die Qualitäten der Algorithmen wird im Kapitel 4 diskutiert.

Kapitel 4.

Diskussion der Ergebnisse

In diesem Kapitel werden die drei Algorithmen anhand der Tabelle 3.3 und den Abbildungen 3.14 (a) und (b) untereinander verglichen. Dabei wird der Vergleich in folgende Kategorien:

- Menge der Datenpunkte,
- Anzahl der Iterationen,
- Genauigkeit,
- Wahl des Startpunktes,

unterteilt. Die Komplexität der Implementierung soll außer Acht gelassen werden.

Menge der Datenpunkte

Die ersten zwei Algorithmen nehmen zur Auswertung einen Datenpunkt , in dem konkreten Fall das Maximum der Messung, bzw. der analytischen Lösung. Der dritte Algorithmus benutzt im Gegenteil alle Datenpunkte und kann damit exakter komplizierte Verläufe angleichen. In dieser Hinsicht ist die Anwendung des letzten Algorithmus effizienter, als bei nichtlinearen Verläufen. Wie z.B. die experimentelle Messkurven in der Bruchmechanik.

Anzahl der Iterationen

Wie im Kapitel 3.1 Abschnitt 3.2 erwähnt, basieren die Schrittweitensteuerung der Algorithmen RS und RSR auf den Pseudo-Zufallszahlengeneratoren, so dass die Anzahl der Iterationen bei mehreren Durchläufen mit gleichen Werten unterschiedliche Anzahl der Iterationen liefern kann. Aus der Tabelle 3.3 lässt sich ablesen, dass Diff Algorithmus mit zwei Iterationen deutlich vorne liegt. Der RSR Algorithmus hat zwar nach 15 Iterationen konvergiert, aber im Vergleich mit RS Algorithmus, der ebenfalls pseudo-zufallsbasiert ist, jedoch 2,6 mal schneller.

Genauigkeit

Zur Beurteilung der Genauigkeit sei der relative Fehler (siehe Abbildung 3.14), bezogen auf den exakten Wert $E = 2,1e+05\ MPa$, herangezogen. Der RSR Algorithmus ist nach der Abbildung rechnet zwar mit der größten Abweichung, allerdings liegt der Fehler im Promillebereich, was mehr als akzeptabel ist.

Wahl des Startpunktes

Die Wahl des Startpunktes entscheidet oft in der Optimierung wie lange man bis zu dem Optimum braucht und ob man es überhaupt erreicht. In dieser Arbeit entscheidet die Wahl des Startparameters bei Algorithmen RS und RSR auf die Anzahl der Iterationen bis zu dem Erreichen des Minimums. Wie bereits im Kapitel 3.1 Abschnitt 3.2 erwähnt braucht der RSR Algorithmus bei einem Startwert $E = 1000\ MPa$ schon 319 Iterationen bis zum Optimums. Der Diff Algorithmus konvergiert dagegen, zumindest für das vorliegende Problem, immer mit zwei Iterationen.

Resümee

Zum Schluss wird versucht eine Aussage zu treffen, welcher Algorithmus mit dem Problem am besten umgeht. Resümiert man das Kapitel, so kann man eine Bewertungsmatrix, die man in der Tabelle 4.1 sieht, aufstellen. Das Kreuz "×"markiert den Algorithmus, der am effizientesten die Aufgabe gelöst hat.

	Menge der Datenpunkte	Anzahl der Iterationen	Genauigkeit	Wahl des Startpunktes
RS Algorithmus				
Diff Algorithmus		×	×	×
RSR Algorithmus	×			

Tabelle 4.1.: Bewertungsmatrix für Algorithmen.

Laut der Tabelle 4.1 kann man den RS Algorithmus als ineffizientesten wählen. Diff Algorithmus hat zwar die meisten Kreuze, allerdings wird zur Auswertung ein Punkt herangezogen. Der RSR Algorithmus ist zwar langsamer, aber universeller, weil man alle Datenpunkte aus der Messung mitnimmt.

Kapitel 5.

Zusammenfassung und Ausblick

In diesem Kapitel wird vorerst über die gesamte Arbeit resümiert und die wichtigsten Inhalte im Abschnitt 5.1 zusammengefasst. Anschließend diskutiert man im Abschnitt 5.2 über offene Möglichkeiten in darauffolgenden Arbeiten.

5.1. Zusammenfassung

Der Schwerpunkt der vorliegenden Arbeit liegt in der Entwicklung eines Algorithmus zu inversen Parameteridentifikation. Zur Einführung wurden die theoretischen Grundlagen der Bruchmechanik und Optimierung im Kapitel 2 erarbeitet. Die bruchmechanischen Grundlagen dienen dem Verständnis der Problematik. Der Leser bekam einen Überblick über die wichtigsten Methoden und deren Lösungen. Dabei wurde die Notwendigkeit der inversen Parameteridentifikation bei Separationsgesetzen der Kohäsivzonenmodelle aufgezeigt. Im Abschnitt 2.2 sind die bedeutsamsten Optimierungstechniken zusammengefasst. Besonders wurde auf die Methode der kleinsten Quadrate eingegangen, da sie als mathematische Grundlage für die Entwicklung des Algorithmus dient. Zum Schluss des Kapitels 2 hat man allgemeine Imformation zur Implementierung in FE-Programm FEAP zusammengefasst. Auf die Grundlagen der Benutzung des Programms wurde dabei verzichtet. Die Erarbeitung in FEAP soll dem Leser überlassen werden. Und hat lediglich für die Implementierung notwendige Aspekte angesprochen.

Das Kernkapitel 3 beinhaltet die notwendigen Informationen zu Programmstruktur, Modellierung, Implementierung selbst, Validierung und Diskussion der Ergebnisse. Als erstes wurden die allgemeine Programmstruktur und Modellaufbau erläutert. Nach dem die grundlegenden Aspekte der Simulationen geklärt waren, wurden drei entwickelte Algorithmen vorgestellt. Dabei hat man die Funktionsweise jedes einzelnen Algorithmus mit Hilfe von schematischen Diagrammen und den Struktogrammen erklärt. Als nächster Abschnitt folgte die Validierung der Algorithmen. Man hat ein einfaches Modell einer Kragscheibe genommen, das eine geschlossen-analytische Lösung besitzt, um die Validierung möglichst genau durchführen zu können. Zu der besseren Beurteilung der Konvergenz der Algorithmen hat man eine Schnittstelle zu dem Grafikprogramm Gnuplot implementiert. Auf diese Weise ist es dem Nutzer besser erkennbar, ob der Algorithmus konvergiert oder divergiert. Mit Hilfe der Algorithmen sollte das Elastizitätsmodul mit inversen Methoden identifiziert werden. Alle Algorithmen haben unterschiedlich gut das Problem gelöst. RS Algorithmus hat im Vergleich zu den zwei anderen Algorithmen schlechter die Ergebnisse erzielt, weil er

nur einen Punkt auswertet und mehr Iterationen als Diff Algorithmus gebraucht hat. Diff Algorithmus war am effizientesten für einfache Probleme, weil in dem Algorithmus nur ein Punkt und die Differenz zwischen Simulation und Experiment als Faktor ausgewertet wird. RSR Algorithmus war universeller, weil für die Auswertung alle Messpunkte herangezogen werden. Die Diskussion der Ergebnisse und Algorithmen soll, mit der im Kapitel 4 erstellten Bewertungsmatrix, dem Nutzer helfen, den für seine individuelle Problemstellung passenden Algorithmus zu wählen.

Es konnte gezeigt werden, dass die inverse Paramederidentifikation mit dem FE-Programm FEAP erfolgreich umsetzbar ist und liefert plausible Ergebnisse.

5.2. Ausblick

Die vorliegende Arbeit ist wie jede andere Abschlussarbeit zeitlich begrenzt und es ergeben sich viele weitere Fragestellungen. Deswegen wurden einige Ideen aus den zeitlichen Gründen nicht umgesetzt und werden im diesem Abschnitt für die zukünftige Arbeiten erläutert.

Aus der numerischen Sicht wäre es sinnvoll weitere Algorithmen zu implementieren, damit der Nutzer ein breites Spektrum an Werkzeugen hat. Eine passende Schnittstelle zur Einführung von weiteren Optimierungsalgorithmen wurde für solche Zwecke eingebaut. Weiterhin könnten gewisse Abbruchmechanismen im Falle der Divergenz implementiert werden. Es kann sein, dass bei der ungünstigen Wahl der Startparameter die FE-Rechnung nicht konvergiert, beispielsweise ist es der Fall bei den Kohäsivparametern, und man unnötig die Rechenressourcen vergeudet.

Physikalisch ist es wünschenswert die Algorithmen anhand der kohäsiven Parameter δ_c, σ_c und G_C zu testen. Die vorliegende Arbeit und die Masterarbeit von Fulgence [20] sollen als Grundlage für diese Untersuchungen dienen. Die Messkurven sind bei bruchmechanischen Prozessen sind deutlich anspruchsvoller. Darüber hinaus kann man die Algorithmen anhand der weiteren nichtlinearen Berechnungen validieren.

Literatur

[1] **Anderson**, **T. L.** *Fracture mechanics: Fundamentals and applications*. 3. ed. Boca Raton: CRC Taylor & Francis, **2005**. ISBN: 978-0849316562.

[2] **Apel**, **T.** „Numerische Parameteridentifikation“. Skript zur Vorlesung. München: Universität der Bundeswehr München, **2012**.

[3] **Baier**, **H.**, **Seeßelberg**, **C.** und **Specht**, **B.** *Optimierung in der Strukturmechanik*. Wiesbaden: Vieweg+Teubner Verlag, **1994**. ISBN: 978-3-322-90701-1. DOI: 10.1007/978-3-322-90700-4.

[4] **Barenblatt**, **G. I.** „The Mathematical Theory of Equilibrium Cracks in Brittle Fracture“. In: *Advances in Applied Mechanics Volume 7*. Bd. 7. Advances in Applied Mechanics. Elsevier, **1962**, S. 55–129. ISBN: 9780120020072. DOI: 10.1016/S0065-2156(08)70121-2.

[5] **Betten**, **J.** *Finite Elemente für Ingenieure 1: Grundlagen, Matrixmethoden, Elastisches Kontinuum*. Berlin und Heidelberg: Springer, **2003**. ISBN: 978-3-642-62443-8. DOI: 10.1007/978-3-642-55536-7.

[6] **Blobel**, **V.** und **Lohrmann**, **E.** *Statistische und numerische Methoden der Datenanalyse*. Teubner Studienbücher Physik. Wiesbaden: Vieweg+Teubner Verlag, **1998**. ISBN: 978-3-519-03243-4. DOI: 10.1007/978-3-663-05690-4.

[7] **Blumenauer**, **H.** und **Pusch**, **G.** *Technische Bruchmechanik*. 3., stark überarb. Aufl. Leipzig u.a.: Dt. Verl. für Grundstoffindustrie, **1993**. ISBN: 978-3342006596.

[8] **Brinson**, **L.C.** und **Lammering**, **R.** „Finite element analysis of the behavior of shape memory alloys and their applications“. In: *International Journal of Solids and Structures* 30.23 **(1993)**, S. 3261–3280. ISSN: 00207683. DOI: 10.1016/0020-7683(93)90113-L.

[9] **Burdekin**, **F. M.** und **Stone**, **D. E. W.** „The crack opening displacement approach to fracture mechanics in yielding materials“. In: *The Journal of Strain Analysis for Engineering Design* 1.2 **(1966)**, S. 145–153. DOI: 10.1243/03093247V012145.

[10] **Campilho**, **R.D.S.G.** u. a. „Modelling adhesive joints with cohesive zone models: effect of the cohesive law shape of the adhesive layer“. In: *International Journal of Adhesion and Adhesives* 44 **(2013)**, S. 48–56. ISSN: 01437496. DOI: 10.1016/j.ijadhadh.2013.02.006.

[11] **Cherepanov**, **G.P.** „Crack propagation in continuous media“. In: *Journal of Applied Mathematics and Mechanics* 31.3 **(1967)**, S. 503–512. ISSN: 00218928. DOI: 10.1016/0021-8928(67)90034-2.

[12] **Cornec**, **A.**, **Scheider**, **I.** und **Schwalbe**, **K.-H.** „On the practical application of the cohesive model“. In: *Engineering Fracture Mechanics* 70.14 **(2003)**, S. 1963–1987. ISSN: 00137944. DOI: 10.1016/S0013-7944(03)00134-6.

[13] **da Silva, L. F. M.** und **Campilho, R. D. S. G.** *Advances in Numerical Modeling of Adhesive Joints*. SpringerBriefs in Applied Sciences and Technology. Berlin, Heidelberg: Springer Berlin Heidelberg, **2012**, S. vi + 151. ISBN: 978-3-642-23607-5. DOI: 10.1007/978-3-642-23608-2.

[14] **Dahmen, W.** und **Reusken, A.** *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, **2008**. ISBN: 978-3-540-76492-2. DOI: 10.1007/978-3-540-76493-9.

[15] **Darcis, Ph. P.** u. a. „Crack Tip Opening Angle: Measurement and Modeling of Fracture Resistance in Low and High Strength Pipeline Steels". In: *Volume 3: Materials and Joining; Pipeline Automation and Measurement; Risk and Reliability, Parts A and B*. Bd. 2006. ASME, **2006**, S. 159–168. ISBN: 0-7918-4263-0. DOI: 10.1115/IPC2006-10172.

[16] **Dugdale, D. S.** „Yielding of steel sheets containing slits". In: *Journal of the Mechanics and Physics of Solids* 8.2 **(1960)**, S. 100–104. ISSN: 00225096. DOI: 10.1016/0022-5096(60)90013-2.

[17] **Elices, M.** u. a. „The cohesive zone model: Advantages, limitations and challenges". In: *Engineering Fracture Mechanics* 69.2 **(2002)**, S. 137–163. ISSN: 00137944. DOI: 10.1016/S0013-7944(01)00083-2.

[18] **Frediani, A.** „Experimental measurement of the J-integral". In: *Engineering Fracture Mechanics* 19.6 **(1984)**, S. 1105–1137. ISSN: 00137944. DOI: 10.1016/0013-7944(84)90155-3.

[19] **Freund, R. W.** und **Hoppe, R. H.W.** *Stoer/Bulirsch: Numerische Mathematik 1*. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, **2007**. ISBN: 978-3-540-45389-5. DOI: 10.1007/978-3-540-45390-1.

[20] **Fulgence, Nikiema Wendwoga**. „Implementierung eines Algorithmus zur Beschreibung kohäsiven Materialversagens". Masterarbeit. Helmut - Schmidt - Universität, Professur für Mechanik, **2016**.

[21] **Gander, W.** *Computermathematik*. Bd. 3. Basel und s.l.: Birkhäuser Basel, **1992**. ISBN: 978-3-7643-2765-1. DOI: 10.1007/978-3-0348-5671-3.

[22] **Geiger, C.** und **Kanzow, C.** *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer-Lehrbuch. Berlin und Heidelberg: Springer, **1999**. ISBN: 978-3-540-66220-4. DOI: 10.1007/978-3-642-58582-1.

[23] **Gekeler, E. W.** *Mathematische Methoden zur Mechanik*. Berlin und Heidelberg: Springer-Verlag Berlin Heidelberg, **2010**. ISBN: 978-3-642-14252-9. DOI: 10.1007/978-3-642-14253-6.

[24] **Giurgiutiu, V.** *Structural health monitoring with piezoelectric wafer active sensors*. Amsterdam: Academic Press/Elsevier, **2008**. ISBN: 978-0-12-088760-6. DOI: 10.1016/b978-0-12-088760-6.x5001-6.

[25] **Gratton, S., Lawless, A.S.** und **Nichols, N.K.** „Approximate Gauss-Newton methods for nonlinear least squares problems". Numerical Analysis Report. The University of Reading, Department of Mathematics, **2004**.

[26] **Gross**, **D.** und **Seelig**, **Th.** *Bruchmechanik: Mit einer Einführung in die Mikromechanik*. 6. Aufl. 2016. Berlin, Heidelberg und s.l.: Springer Berlin Heidelberg, **2016**. ISBN: 978-3-662-46736-7. DOI: 10.1007/978-3-662-46737-4.

[27] **Hahn**, **H. G.** *Wege und Ziele der Konzepte in der Bruchmechanik: Technische Mechanik 8, Heft 1*. **1986**.

[28] **Heinrich**, **F.** und **Lammering**, **R.** „A numerical approach to simulate printed electronics on fiber reinforced composites“. In: *PAMM* 16.1 **(2016)**, S. 135–136. ISSN: 16177061. DOI: 10.1002/pamm.201610056.

[29] **Hennecke**, **M.** „A Fortran 90 interface to random number generation“. In: *Computer Physics Communications* 90.1 **(1995)**, S. 117–120. ISSN: 00104655. DOI: 10.1016/0010-4655(95)00065-N.

[30] **Hillerborg**, **A.**, **Modéer**, **M.** und **Petersson**, **P.-E.** „Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements“. In: *Cement and Concrete Research* 6.6 **(1976)**, S. 773–781. ISSN: 00088846. DOI: 10.1016/0008-8846(76)90007-7.

[31] **Huckle**, **T.** und **Schneider**, **S.** *Numerik für Informatiker*. Berlin und Heidelberg: Springer, **2002**. ISBN: 978-3-540-42387-4. DOI: 10.1007/978-3-662-09019-0.

[32] **Hui**, **C. Y.** u. a. „Cohesive Zone Models and Fracture“. In: *The Journal of Adhesion* 87.1 **(2011)**, S. 1–52. ISSN: 0021-8464. DOI: 10.1080/00218464.2011.538315.

[33] **Jarre**, **F.** und **Stoer**, **J.** *Optimierung*. Springer-Lehrbuch. Berlin und Heidelberg: Springer, **2004**. ISBN: 978-3-540-43575-4. DOI: 10.1007/978-3-642-18785-8.

[34] **Jung Lee**, **M.** u. a. „Determination of cohesive parameters for a mixed-mode cohesive zone model“. In: *International Journal of Adhesion and Adhesives* 30.5 **(2010)**, S. 322–328. ISSN: 01437496. DOI: 10.1016/j.ijadhadh.2009.10.005.

[35] **Kaiser**, **B.** *Implementierung einer Methode zur automatischen 3D-FEM Modellerstellung und Festigkeitsrechnung für Vollhartmetall-Spiralbohrer: Unter Verwendung von nicht kommerzieller Matlab-Software*. Diplomica Verlag, **2011**.

[36] **Kernighan**, **B. W.** und **Ritchie**, **D. M.** *The C programming language*. 2nd. **1988**. ISBN: 0201889544.

[37] **Knothe**, **K.** und **Wessels**, **H.** *Finite Elemente: Eine Einführung für Ingenieure*. **1999**. ISBN: 978-3-662-07236-3. DOI: 10.1007/978-3-662-07235-6.

[38] **Kókai**, **G.** „Erfolge und Probleme evolutionärer Algorithmen, induktiver logischer Programmierung und ihrer Kombination“. Diss. Universität Erlangen-Nürnberg, Institut für Informatik, **2003**.

[39] **Kolinksy**, **H.** „Programmieren in Fortran 90/95“. Skript. Universität Bayreuth, **2012**.

[40] **Kozicki**, **J.** und **Donzé**, **F. V.** „A new open-source software developed for numerical simulations using discrete modeling methods“. In: *Computer Methods in Applied Mechanics and Engineering* 197.49-50 **(2008)**, S. 4429–4443. ISSN: 00457825. DOI: 10.1016/j.cma.2008.05.023.

[41] **Kuna**, **M.** *Numerische Beanspruchungsanalsye von Rissen: Finite Elemente in der Bruchmechanik*. 1. Aufl. Vieweg+Teubner (GWV), **2008**. ISBN: 978-3-8351-0097-8. DOI: 10.1007/978-3-8348-9285-0.

[42] **Lawson**, **C. L.** und **Hanson**, **R. J.** *Solving Least Squares Problems*. Society for Industrial und Applied Mathematics, **1995**. ISBN: 978-0-89871-356-5. DOI: 10.1137/1.9781611971217.

[43] **Louis**, **A. K.** *Inverse und schlecht gestellte Probleme*. Wiesbaden: Vieweg+Teubner Verlag, **1989**. ISBN: 978-3-519-02084-4. DOI: 10.1007/978-3-322-84808-6.

[44] **Mahler**, **M.** *Entwicklung einer Auswertemethode für bruchmechanische Versuche an kleinen Proben auf der Basis eines Kohäsivzonenmodells*. Bd. 51. Schriftenreihe des Instituts für Angewandte Materialien, Karlsruher Institut für Technologie. Karlsruhe, Baden: KIT Scientific Publishing, **2016**. ISBN: 978-3-7315-0441-2.

[45] **Maier**, **G.** u. a. „Inverse analyses in fracture mechanics“. In: *International Journal of Fracture* 138.1-4 **(2006)**, S. 47–73. ISSN: 03769429. DOI: 10.1007/s10704-006-7153-7.

[46] **Moré**, **J. J.** und **Wright**, **S. J.** *Optimization Software Guide*. Society for Industrial und Applied Mathematics, **1993**. ISBN: 978-0-89871-322-0. DOI: 10.1137/1.9781611970951.

[47] **Nocedal**, **J.** und **Wright**, **S. J.** *Numerical optimization*. Corr. 2. print. Springer series in operations research. New York, NY: Springer, **2000**. ISBN: 0-387-98793-2.

[48] **Press**, **W. H.** u. a. *Numerical Recipes in FORTRAN; The Art of Scientific Computing*. New York, NY, USA, **1993**.

[49] **Rice**, **J. R.** „A Path Independent Integral and the Approximate Analysis of Strain Concentration by Notches and Cracks“. In: *Journal of Applied Mechanics* 35 **(1968)**, S. 379–386.

[50] **Rossmanith**, **H.-P.** *Grundlagen der Bruchmechanik*. Vienna: Springer Vienna, **1982**. ISBN: 978-3-7091-8648-0. DOI: 10.1007/978-3-7091-8647-3.

[51] **Sakhalkar**, **A.** u. a. „Crack Tip Opening Angle Measurement Methods and Crack Tunnelling in 2024-T351 Aluminium Alloy“. In: *Strain* 47 **(2011)**, e130–e141. ISSN: 00392103. DOI: 10.1111/j.1475-1305.2008.00579.x.

[52] **Scheider**, **I.** und **Brocks**, **W.** „The Effect of the Traction Separation Law on the Results of Cohesive Zone Crack Propagation Analyses“. In: *Key Engineering Materials* 251-252 **(2003)**, S. 313–318. ISSN: 1662-9795. DOI: 10.4028/www.scientific.net/KEM.251-252.313.

[53] **Schwalbe**, **K.-H.**, **Scheider**, **I.** und **Cornec**, **A.** *Guidelines for Applying Cohesive Models to the Damage Behaviour of Engineering Materials and Structures*. SpringerBriefs in Applied Sciences and Technology. Berlin, Heidelberg: Springer, **2013**. ISBN: 978-3-642-29494-5. DOI: 10.1007/978-3-642-29494-5.

[54] **Schwarz**, **H. R.** und **Köckler**, **N.** *Numerische Mathematik*. Wiesbaden: Vieweg+Teubner, **2009**. ISBN: 978-3-8348-0683-3. DOI: 10.1007/978-3-8348-9282-9.

[55] **Stigh**, **U.**, **Alfredsson**, **K. S.** und **Biel**, **A.** „Measurement of Cohesive Laws and Related Problems“. In: *Volume 11: Mechanics of Solids, Structures and Fluids*. ASME, **2009**, S. 293–298. ISBN: 978-0-7918-4384-0. DOI: 10.1115/IMECE2009-10474.

[56] **Törnig**, **W.** und **Spellucci**, **P.** *Numerische Mathematik für Ingenieure und Physiker 1*. Berlin, Heidelberg: Springer Berlin Heidelberg, **1988**, S. 353. ISBN: 978-3-540-19192-6. DOI: 10.1007/978-3-642-87671-4.

[57] **Weisskopf**, **D.** „Untersuchung, Implementierung und Einsatz des Levenberg-Marquardt-Algorithmus zur Analyse von Chip-Fertigungsdaten“. Diplomarbeit. Eberhard-Karls-Universität Tübingen, Fachbereich Informatik - Angewandte Informatik, **1999**.

[58] **Wells**, **A.A.** „Unstable crack propagation in metals: cleavage and fast fracture“. In: *Proceedings of the crack propagation symposium*. Bd. 1. 84. **1961**.

[59] **Wissmann**, **J.** und **Sarnes**, **K.-D.** *Finite Elemente in der Strukturmechanik*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, **2006**. ISBN: 978-3-540-61836-2. DOI: 10.1007/3-540-29277-2.

[60] **Xie**, **D.** u. a. „Cohesive Zone Model for Surface Cracks Using Finite Element Analysis“. In: *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 16th AIAA/ASME/AHS Adaptive Structures Conference 10t*. Reston, Virigina: American Institute of Aeronautics und Astronautics, **2008**. ISBN: 978-1-60086-993-8. DOI: 10.2514/6.2008-1764.

[61] **Zienkiewicz**, **O. C.** und **Taylor**, **R. L.** *The finite element method. Volume 1: The Basis*. 5. ed., reprinted. Oxford: Butterworth-Heinemann, **2002**. ISBN: 0-7506-5049-4. DOI: 10.1002/bate.200201090.

[62] **Zienkiewicz**, **O.C.**, **Taylor**, **R.L.** und **Fox**, **D.** *The Finite Element Method for Solid and Structural Mechanics*. Elsevier, **2014**, S. 215–233. ISBN: 9781856176347. DOI: 10.1016/B978-1-85617-634-7.00007-7.

[63] **Zienkiewicz**, **O.C.**, **Taylor**, **R.L.** und **Zhu**, **J.Z**. *The Finite Element Method: Its Basis and Fundamentals*. **2013**, S. 756. ISBN: 9781856176330.

Anhang A.

FEAP- und Fortran-Code

A.1. FEAP Input Dateien

```
PARAmeter
  k1 = 0.7E+05            ! Startparameter zur Initialisierung
! Leere Zeile
LOOP,100

INCLude Ischeibe
  POPT,half,force,k1,y105,data.txt
NEXT
! Leere Zeile
END
! Leere Zeile
INTEractive
! Leere Zeile
STOP
```

Code A.1: FEAP `main`-Datei.

```
FEAP
0   0   0   2   2   4
! Leere Zeile
MATErial 1
SOLID
ELASTIC ISOTROPIC  k1    0.3
PLANE STREss
THICk,,1,0,0
! Leere Zeile
PARAm
n2 = 20
n1 = 4
d1 = -12.0
l = 100.0
b = 20.0
it = 1        ! Anzahl der Iterationen
f0 = 1000.0   ! Gesamtkraft
f1 = f0/it    ! Gesamtkraft/Anzahl der Iterationen = Kraftstep
! Leere Zeile
```

```
BLOCK
CARTESIAN n2 n1
1   0.0   0.0
2   l     0.0
3   l     b
4   0.0   b
! Blank line
ebou
1 0.0 1 1 1
! Leere Zeile
CFOR
NODE 100 20 0 f1
! Leere Zeile
END
! Blank line
BATCh
  tang,,1
  PLOT POSTscript
  plot cont 2
  plot boun
  PLOT MESH
  PLOT LOAD,,-1
  PLOT node
  PLOT POSTscript
  plot,DEFO
  outp, ndis
  plot wipe
  plot fill
  plot cont 2
  reac,all
  outp,nfrc
! Blank line
END
```

Code A.2: FEAP `Ifile`-Datei.

A.2. Fortran Codes

```
      SUBROUTINE popt(AlgoName,Lasttyp,ParameterName,Knoten,
     DateiName)
      use optimierung

      implicit none

      include   'iofile.h'
      include   'pointer.h'
      include   'comblk.h'
      include   'iodata.h'
      include   'sdata.h'
      include   'cdata.h'

      character :: DateiName*(*),
     &             ParameterName*(*),
     &             Knoten*(*),
     &             AlgoName*(*),
     &             Lasttyp*(*)

      integer :: io,i,j,knotint,nodedirec

      logical :: pcomp

      logical  :: error,prt,mess_vorhanden
      real*8   :: val,val2,disp,reac,korpar
      real,parameter :: tol = 1e-05
      real,parameter :: Toleranz = 0.0001

      save

      inquire(file=DateiName,exist=mess_vorhanden)

      if (.not.mess_vorhanden) then
          write(*,*) "-> Keine Messdaten vorhanden !"
          write(*,*) "   FEAP wird beendet."
          write(*,*) "   Enter druecken..."
          read(*,*)
          call plstop
      end if

      nrcols = 2
      nrrows = nrrow(Dateiname)

      call knToint(Knoten,knotint,nodedirec)
      call nodevalues(reac,disp,nodedirec,knotint)

      allocate(measMatrix(nrrows,nrcols),stat=alloc_status)
```

```
47          call messmat(Dateiname,nrrows,nrcols,measMatrix,writeDat)
48
49        allocate(SimVekU(nrrows))
50        allocate(SimVekF(nrrows))
51        allocate(messVekU(nrrows))
52        allocate(messVekF(nrrows))
53
54        do i=1,nrrows
55            messVekU(i) = measMatrix(i,2)
56        end do
57
58        call diskrData(measMatrix(:,1),SimVekU,disp)
59
60        call diskrData(measMatrix(:,2),SimVekF,reac)
61
62        if (pcomp(Lasttyp,"forc",4)) then
63            if (pcomp(AlgoName(1:4),"rand",4)) then
64                call algo_random(ParameterName,val,val2,
65       &             measMatrix(nrrows,2),disp,tol,
66       &             Knoten, DateiName,reac,disp,measMatrix,nrrows,
         nrcols,
67       &             SimVekF,SimVekU)
68            elseif(pcomp(AlgoName(1:4),"diff",4)) then
69                call algo_diff(ParameterName,val,val2,
70       &             measMatrix(nrrows,2),disp,1e-06,
71       &             Knoten, DateiName,reac,disp,measMatrix,nrrows,
         nrcols,
72       &             SimVekF,SimVekU)
73            elseif(pcomp(AlgoName(1:4),"rstp",4)) then
74                call algo_rnd_stps(ParameterName,val,val2,
75       &             messVekU,SimVekU,Toleranz,
76       &             Knoten, DateiName,reac,disp,measMatrix,nrrows,
         nrcols,
77       &             SimVekF,SimVekU)
78            endif
79
80        else if (pcomp(Lasttyp,"disp",4)) then
81
82            if (pcomp(AlgoName(1:4),"rand",4)) then
83                call algo_random(ParameterName,val,val2,
84       &             measMatrix(nrrows,1),reac,tol,
85       &             Knoten, DateiName,reac,disp,measMatrix,nrrows,
         nrcols,
86       &             SimVekF,SimVekU)
87            elseif(pcomp(AlgoName(1:4),"diff",4)) then
88                call algo_diff(ParameterName,val,val2,
89       &             measMatrix(nrrows,1),reac,1e-06,
90       &             Knoten, DateiName,reac,disp,measMatrix,nrrows,
         nrcols,
```

```
91      &              SimVekF,SimVekU)
92           elseif(pcomp(AlgoName(1:4),"rstp",4)) then
93               call algo_rnd_stps(ParameterName,val,val2,
94      &              messVekF,SimVekF,Toleranz,
95      &              Knoten, DateiName,reac,disp,measMatrix,nrrows,
    nrcols,
96      &              SimVekF,SimVekU)
97           endif
98       !else if (pcomp(Lasttyp,"temp",4)) then
99
100      end if
101
102
103          write(iow,"(/,/)")
104          write(iow,99) "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
105     &~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
106          write(iow,99) "     Werte waherend der Optimierung"
107          write(iow,99) "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
108     &~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
109
110       !---------------------------------!
111       !       Eingabeparameter          !
112       !---------------------------------!
113          write(iow,88) "E i n g a b e p a r a m e t e r:"
114          write(iow,"(A18,1X,A4)") "Parameter = ", ParameterName
115          write(iow,"(A18,1X,A)") "Knoten = ", Knoten(2:)
116          write(iow,"(A18,1X,A1)") "Richtung = ", Knoten(:1)
117          write(iow,"(A18,1X,A)") 'Messdaten = ', DateiName
118          write(iow,"(A18,1X,ES8.1)")"Toleranz = ", tol
119
120       !---------------------------------!
121       !           FE-Werte              !
122       !---------------------------------!
123          write(iow,88) "F E - W e r t e:"
124          write(iow,"(A18,1X,F12.1)") "Knotenkraft = ", reac
125          write(iow,"(A18,1X,F12.6)") "Verschiebung = ", disp
126
127       !---------------------------------!
128       !           Messwerte             !
129       !---------------------------------!
130          write(iow,88) "M e s s w e r t e:"
131
132          write(iow,"(21X,A10,4X,A10)") "1. Wert","2. Wert"
133
134          write(iow,"(A18,1X,F12.1,A2,F12.6)") "Maximas = ",
135     &     maxval(measMatrix(:,1)),", ",maxval(measMatrix(:,2))
136
137          write(iow,"(A18,1X,F12.1,A2,F12.6)") "Minimas = ",
138     &     minval(measMatrix(:,1)),", ",minval(measMatrix(:,2))
```

```
139
140           write(iow,"(/,A11,30X,A11)") "Messdaten: ", "FE-Daten: "
141           write(iow,"(A18,6X,A10,7X,A15,12X,A10)")
142      &     "1. Wert","2. Wert","F diskr","U diskr"
143
144           do i = 1,nrrows
145             write(iow,"(F18.9,4X,F12.9,4X,F18.9,4X,F18.9)")
146      &           (measMatrix(i,j), j = 1,nrcols),SimVekF(i),SimVekU
    (i)
147           end do
148
149           write(iow,88) "R e s i d u e n:"
150           write(iow,"(A18,4X,A12,4X,A18,10X,A12)")
151      &    "Residuen F","Residuen^2 F", "Residuen U","Residuen^2 U"
152           do i = 1,nrrows
153               write(iow,"(F18.9,4X,F12.9,4X,F18.9,4X,F18.9)")
154      &        abs(SimVekF(i)-measMatrix(i,1)),
155      &        abs(SimVekF(i)-measMatrix(i,1))**2,
156      &        abs(SimVekU(i)-measMatrix(i,2)),
157      &        abs(SimVekU(i)-measMatrix(i,2))**2
158           end do
159           write(iow,99) "----------------------------------------
160      &-----------------------------------"
161           write(iow,"(A5,1X,F12.6,4X,F12.6,10X,F12.6,10X,F12.6)")
162      &    "sum =",
163      &    sum(abs(SimVekF-measMatrix(:,1))),
164      &    sum(abs(SimVekF-measMatrix(:,1))**2),
165      &    sum(abs(SimVekU-measMatrix(:,2))),
166      &    sum(abs(SimVekU-measMatrix(:,2))**2)
167
168           write(iow,*)
169           write(iow,99) "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
170      &~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
171           write(iow,99) "    Ende des Blocks"
172           write(iow,99) "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
173      &~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
174
175  88    format(/,A,/)
176  99    format(A,/)
177
178
179
180       deallocate(measMatrix,SimVekU,SimVekF,messVekU,messVekF)
181
182       END SUBROUTINE
```

Code A.3: Datei `main.f90`

```
1       SUBROUTINE diskrData(MessVektor,SimVektor,SimWert)
2
```

```
      IMPLICIT NONE

      real*8,dimension(:)                  :: MessVektor,SimVektor
      real*8,allocatable,dimension(:)      :: Faktor
      integer                              :: i
      real*8                               :: SimWert

      allocate(Faktor(size(MessVektor)))

      Faktor =  MessVektor/maxval(MessVektor)

      SimVektor = Faktor * SimWert

      deallocate(Faktor)

      END SUBROUTINE


      DOUBLE PRECISION FUNCTION SumResQuadr(MessVek,SimVek)

      IMPLICIT NONE

      real*8,dimension(:)     :: MessVek,SimVek

      SumResQuadr = sum((MessVek-SimVek)**2)

      END FUNCTION
```

Code A.4: Subroutine zur Diskretisierung der Simulationswertes.

```
      SUBROUTINE gnuplotter(Befehl1,Dataset)
      !-------------------------------------------------------!
      !        Routine zum Steuern der Plotts im GNUPLOT.-----!
      !       Befehl1: Steuert die Plotts                     !
      !       Dataset: Messdaten als externe Datei            !
      !-------------------------------------------------------!
      use plotter

      implicit none

      include 'iodata.h'
      include 'iofile.h'

      logical                 :: vorhanden
      character(len = *)      :: Befehl1, Dataset

      inquire(file=Dataset,exist=vorhanden)

      if (vorhanden) then
```

```
21          select case (Befehl1)
22          case ("messdata")
23              call gnuCodeMessdata(Dataset,GnuDatNam,"Messdaten")
24              call system(syscommand //" "// GnuDatNam)
25              close(unit = gnu_unit,status='delete')
26          case ("simess")
27              call gnuMessSimData(Dataset,"simvek.txt",
28     &           GnuDatNam,"Simulations- und Messwerte")
29              call system(syscommand //" "// GnuDatNam)
30              close(unit = gnu_unit,status='delete')
31          case default
32              write(*,111) "Falsche GNU Eingabe..."
33          end select
34       else
35          write(*,111) "Keine Datei mit Messwerten vorhanden."
36       end if
37
38  111   format(A)
39       END SUBROUTINE
```

Code A.5: Subroutine zum Benutzen von Gnuplot.

```
1        SUBROUTINE algo_random(ParameterName,val,val2,umess,usim,
    Toleranz,
2       &            Knoten, Dateiname,reac,disp,measMatrix,nrrows,
    nrcols,
3       &            SimVekF,SimVekU)
4
5        IMPLICIT NONE
6
7        character(len = *)    :: ParameterName
8        real*8                :: val,val2,umess,usim,korpar
9        real                  :: Toleranz
10       logical               :: error,prt
11       integer,parameter     :: n = 2
12       real*8,dimension(n)     :: randnrs
13
14       integer                 :: nrrows, nrcols
15       real*8                  :: reac,disp
16       real*8, dimension(:,:)  :: measMatrix
17       character(len=*)        :: Dateiname, Knoten
18       real*8,dimension(:)     :: SimVekF,SimVekU
19
20       call getparam(ParameterName, val, error, prt)
21       if (abs(umess-usim).gt.Toleranz) then
22           call randArray(randnrs,n)
23           if (abs(umess-usim).gt.Toleranz*1e+04) then
24               korpar = randnrs(2)
25           else if(abs(umess-usim).gt.Toleranz*1e+03) then
26               korpar = randnrs(1)
```

```
27          else if(abs(umess-usim).gt.Toleranz*1e+02) then
28              korpar = randnrs(1)*randnrs(1)
29          else if(abs(umess-usim).gt.Toleranz) then
30              korpar = randnrs(1)*randnrs(1)*randnrs(1)
31          end if
32
33          if (umess.lt.usim) then
34              val2 = val + val*korpar
35              call setparam(ParameterName, val2, prt)
36              write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
37          else
38              val2 = val - val*korpar
39              call setparam(ParameterName, val2, prt)
40              write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
41          end if
42       else
43          call output(ParameterName,Knoten,Dateiname,Toleranz,reac,
    disp,
44       &         measMatrix,nrrows,nrcols,val2,
45       &                     SimVekF,SimVekU)
46          write(*,99) "Optimierung ist abgeschlossen."
47          write(*,99) "Loesung:"
48          write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
49          write(*,99)
50       &     "Naehere Information sind in ParamOutput.txt zu finden"
51          write(*,99) "Zum Beenden Enter druecken..."
52          read(*,*)
53          call plstop()
54       end if
55  99   format(/,3X,A)
56       END SUBROUTINE
```

Code A.6: Subroutine mit dem RS Algorithmus.

```
1        SUBROUTINE algo_diff(ParameterName,val,val2,val_mess,val_sim,
2       &            Toleranz,
3       &            Knoten, Dateiname,reac,disp,measMatrix,nrrows,
    nrcols,
4       &            SimVekF,SimVekU)
5
6        IMPLICIT NONE
7
8        character(len = *)   :: ParameterName
9        real*8               :: val,val2,val_mess,val_sim,korpar
10       real                 :: Toleranz
11       logical              :: error,prt
12       real*8               :: faktor
13
14       integer                  :: nrrows, nrcols
15       real*8                   :: reac,disp
```

```
      real*8, dimension(:,:)    :: measMatrix
      character(len=*)          :: Dateiname, Knoten
      real*8,dimension(:)       :: SimVekF,SimVekU

      call getparam(ParameterName, val, error, prt)
      if (abs(val_mess-val_sim).gt.Toleranz) then
          faktor = (val_sim-val_mess)/val_mess
          if (val_mess.lt.val_sim) then
              val2 = val + val*faktor
              call setparam(ParameterName, val2, prt)
              write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
          else
              val2 = val - val*faktor
              call setparam(ParameterName, val2, prt)
              write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
          end if
      else
          call output(ParameterName,Knoten,Dateiname,Toleranz,reac,
   disp,
     & measMatrix,nrrows,nrcols,val2,SimVekF,SimVekU)
          write(*,99) "Optimierung ist abgeschlossen."
          write(*,99) "Loesung:"
          write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val2
          write(*,99)
     &     "Naehere Information sind in ParamOutput.txt zu finden"
          write(*,99) "Zum Beenden Enter druecken..."
          read(*,*)
          call plstop()
       end if
99     format(/,3X,A)
       END SUBROUTINE
```

Code A.7: Subroutine mit dem Diff Algorithmus.

```
      SUBROUTINE algo_rnd_stps(ParameterName,val,val2,val_mess,
   val_sim,
     &            Toleranz,
     &            Knoten, Dateiname,reac,disp,measMatrix,nrrows,
   nrcols,
     &            SimVekF,SimVekU)

      IMPLICIT NONE

      include "iodata.h"

      character(len = 2)        :: ParameterName
      real*8                    :: val,val2
      real                      :: Toleranz
      logical                   :: error,prt
      real*8                    :: faktor
```

```
15
16      integer                     :: nrrows, nrcols
17      real*8                      :: reac,disp
18      real*8, dimension(:,:)      :: measMatrix
19      character(len=*)            :: Dateiname, Knoten
20      real*8,dimension(:)         :: SimVekF,SimVekU,val_mess,val_sim
21
22      real*8                      :: SSR2,res1,res2
23      logical                     :: vorhanden
24      integer                     :: AnzRes,i
25      real*8,allocatable,dimension(:)     :: residuen
26      integer,parameter     :: n = 2
27      real*8,dimension(n)     :: randnrs
28
29      call getparam(ParameterName, val, error, prt)
30
31      inquire(file="ssr.txt",exist=vorhanden)
32
33      call randArray(randnrs,n)
34
35      SSR2 = SumResQuadr(val_mess,val_sim)
36
37      if (vorhanden) then
38          AnzRes = nrrow("ssr.txt")
39          allocate(residuen(AnzRes))
40
41          open(unit = 111, file="ssr.txt",status="old",action="read"
    )
42
43          do i=1,AnzRes
44              read(111,*) residuen(i)
45          end do
46
47              res1 = residuen(AnzRes)
48              res2 = SSR2
49
50          if (res1.gt.res2) then
51              val2 = val + val*randnrs(1)/2.0
52          else if(res1.lt.res2) then
53              val2 = val - val*randnrs(1)/2.0
54          else
55              val2 = val
56          end if
57
58      else
59
60          val2 = val + val
61
62      end if
```

```
63
64        close(unit=111)
65
66        open(unit = ios, file = "simvek.txt")
67
68            do i=1,size(val_sim)
69                write(ios,*) SimVekF(i),val_sim(i)
70            end do
71
72        close(unit = ios)
73
74
75        if (SSR2.gt.Toleranz) then
76            open(unit = 999,file= "ssr.txt",position="append")
77                write(999,*) SSR2
78
79        else
80
81            call output(ParameterName,Knoten,Dateiname,Toleranz,reac,
    disp,
82       & measMatrix,nrrows,nrcols,val,SimVekF,SimVekU)
83
84            write(*,99) "Optimierung ist abgeschlossen."
85            write(*,99) "Loesung:"
86            write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val
87            write(*,99)
88       &     "Naehere Information sind in ParamOutput.txt zu finden"
89            write(*,99) "Zum Beenden Enter druecken..."
90
91            close(unit = 999,status="delete")
92
93            read(*,*)
94            call plstop()
95
96        end if
97
98  99    format(/,3X,A)
99
100
101       write(*,"(/,3X,A4,A1,ES16.4)") ParameterName,"=",val
102       call setparam(ParameterName, val2, prt)
103
104       END SUBROUTINE
```

Code A.8: Subroutine mit dem RSR Algorithmus.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Kuzmin

Vorname: Serhiy

dass ich die vorliegende Masterarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Implementierung eines Optimierungsalgorithmus zur inversen Parameteridentifikation von Kohäsivzonenmodellen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Masterarbeit ist erfolgt durch:

Hamburg		
Ort	Datum	Unterschrift im Original