Ausarbeitung von Marcel Bienia (209744) für das Proseminar „Informatik trifft Maschinenbau“ bei Prof.-Ing. Dr. P. Wiederkehr zu dem Thema:

“Evolutionary Optimization of the Failure Behaviour of Load Introduction Elements Integrated During FRP Sandwich Structure Manufacturing”

VON JAN SCHWENNEN, LUKAS KALBHENN, JÉRÔME KLIPFEL, JENS PFEIFLE, DANIEL KUPZIK, JÜRGEN FLEISCHER

11.03.2020

Inhalt

1 Einleitung

In aktuellen Anwendungen der Auto- und Flugindustrie steigen die Anforderungen an die dort genutzten Materialien. Durch größer, stärker und schneller werdende Fahr- und Flugzeuge müssen verwendete Materialien höhere Lasten tragen können und gleichzeitig leicht sein, um CO2-Ausstoß und Fahr- / Flugverhalten zu optimieren [4]. Ein Werkstoff, der diese Anforderungen erfüllt, ist kohlenstofffaserverstärkter Kunststoff (CFK), welcher umgangssprachlich auch als „Carbon“ bekannt ist. CFK ist ein Verbundwerkstoff, dessen Kunststoff-Matrix zwischen zwei Kohlenstofffaserschichten eingebettet ist. Dadurch entsteht ein sehr leichtes und zugleich sehr belastbares Material. Allerdings hat CFK die Problematik, dass es unter großer Krafteinwirkung zu Brechen neigt [2]. Um diesem Verhalten entgegenzuwirken, werden an Verschraubungspunkten oder anderen Stellen, von denen bereits bekannt ist, dass dort eine große Kraft anliegen wird, Einsätze („Inserts“) in das CFK eingelassen. Auf Grundlage von Johansson, 2008 [7] und J. Schwennen, 2016 [6] zeigt sich, dass Einsätze, welche von einer Kunststoffhülse umgeben sind, zu einer Senkung der Produktionskosten und Steigerung der Stabilität führen. Aufgabe der Hülse soll dabei sein die einwirkende Kraft auf feine größere Fläche in dem CFK Bauteil zu verteilen. Bisher ist es sehr aufwendig die Form und Zusammensetzung dieser Einsätze zu bestimmen, da es weder industrielle Standards, noch ein Hilfsmittel zur Bestimmung gibt. Das hier angestrebte Ziel soll sein, ein solches Hilfsmittel zu kreieren. Genauer gesagt, soll ein Framework entstehen, welches aufgrund von eingehenden Krafteinwirkungen eine effiziente und stabile Form und Aufbau eines Inserts ausgibt. Die Auswertung mithilfe des vorgestellten Tools beschränkt sich auf die Optimierung des Kunststoffeinsatzes und nicht des Stahlbolzen (siehe Abb. 1). Erarbeitet werden soll das Framework an einem konkreten Rahmen, welcher eine Stahlschraube durch einen Einsatz in einer CFK Platte befestigt. Daraufhin drückt eine Kraft diese Schraube vertikal in eine Richtung, währenddessen die CFK Platte blockiert ist. Aus einer Vielzahl solcher Versuche soll ein evolutionärer Algorithmus die optimale Geometrie bestimmen.

Abbildung 1 Stahleinsätze

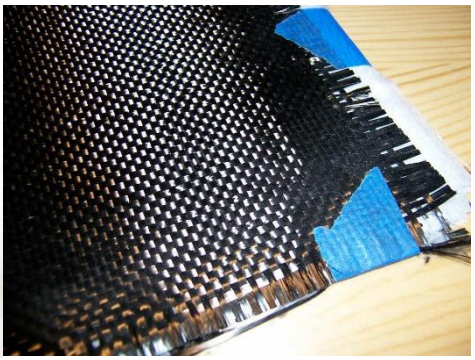


Abbildung 2 Kohlenstofffasergewebe

2 Grundlagen des Frameworks

Im folgendem sollen die Themen, auf denen das Framework basieren wird, grundlegend erläutert werden. Die Zusammenhänge und Anwendungen der Themen werden dann im Teil 3 besprochen.

2.1 Faser Kunststoff Verbund (FKV)

Alle Materialien, welche aus einer Kunststoff Matrix und Verstärkungsfasern bestehen gehören zu der Gruppe der Faser-Kunststoff-Verbunde. In dieser Gruppe werden hohe Festigkeits- und Steifigkeitseigenschaften der Verstärkungsfasern in eine Kunststoff Matrix eingebettet. Diese Kunststoffmatrix verteilt die einwirkende Kraft dann gleichmäßig auf die Fasern, um so ihre Stärke optimal nutzen zu können. Besteht der Werkstoff aus Fasern, die parallel nebeneinander eingebettet sind, so ist dieser Werkstoff nur in Richtung der Faserorientierung steif. Die nebeneinanderliegenden Fasern verhalten sich elastisch. Deshalb werden in den meisten Faserschichten die Fasern 0°/90° versetzt verwebt (siehe Abb. 2).

Die Fasern und Matrizen lassen sich auf viele verschiedene Arten und Weisen kombinieren und können so viele unterschiedliche Materialien erzeugen. Die verschiedenen Fasertypen lassen sich in die Gruppe der anorganischen Fasern und der organischen Fasern zu teilen. Anorganische Fasern sind meist günstig zu produzieren und hitzebeständig. Ein Beispiel dafür sind Glasfasern innerhalb von GFK. Organische Fasern haben oft einen höheren Orientierungsgrad, welcher oftmals zu höherer Steifigkeit des Materials führt. Die bekannteste organische Faser ist die Kohlenstofffaser, welche in Carbon benutzt wird. Die verschiedenen Kunststoff Matrizen lassen sich in erster Linie durch die Art des Kunststoffes unterscheiden. Thermoplasten sind Kunststoffe, die bei Erhitzen verformbar werden, weshalb sich die daraus entstehenden Materialien auch mehrmals verformen lassen. Sie sind jedoch nicht sehr hitzebeständig. Duroplasten sind eine andere Art der Kunststoffe. Diese können nur einmal in Form gebracht werden und sind hitzebeständig.

[2]

2.2 Finite Element Methode (FEM)

Die Finite Element Methode ist ein Verfahren, um physikalische Untersuchungen, wie zum Beispiel Festigkeits- oder Verformungsuntersuchungen auf Basis eines Differentialgleichungssystems numerisch zu lösen. Der Name „Finite-Element“ stammt daher, dass die zu testenden Objekte in viele kleine Elemente, wie z.B. Dreiecke oder Quadrate geteilt werden. Wirkt nun eine physikalische Kraft auf diese Elemente ein, so wird nur die Reaktion der einzelnen Elemente und die Kraftweitergabe an benachbarte Elemente betrachtet. So stellt das Modell eine Annährung des reellen Verhaltens des Objektes dar. Je feiner die Zerteilung ist und je genauer die Randbedingungen, desto genauer ist die Annährung. Allerdings steigt damit auch der benötigte Rechenaufwand zum Lösen des Modells. Nach Lösen des Modells kann das Objekt unter Einwirkung der Kräfte dargestellt werden. Häufig wird dann ein Falschfarbenbild erzeugt, um Verformungen (siehe Abb. 3 A) oder Wärmeverteilungen (siehe Abb. 3 B) darzustellen. [8]

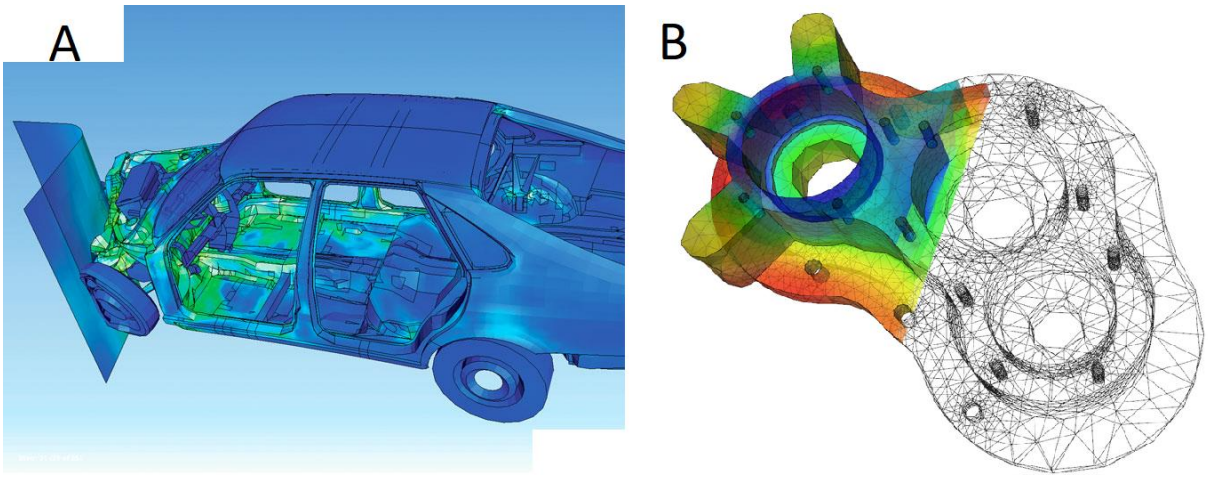


Abbildung 3 (A) FEM Simulation der Verformung eines Autos beim Aufprall / (B) FEM Simulation der Wärmeverteilung in einem Pumpengehäuse

2.3 Evolutionäre Algorithmen (EA)

Evolutionäre Algorithmen bauen auf dem biologischen Konzept der Evolution auf. Dieses Konzept beschreibt die Entwicklung von Lebewesen im Laufe der Zeit, von Generation zu Generation. Laut dieser Theorie entsteht die Entwicklung eines Lebewesens durch ein in jeder Generation wiederkehrendes Verhalten. Wenn eine neue Generation eines Lebewesens entsteht, so sind die Gene eines jedes Individuums eine Kombination aus den Genen der Eltern Individuen (Rekombination). Zusätzlich dazu verändern sich die Gene eines jeden Individuums noch durch Mutation. So entstehen in

jeder Generation neue einzigartige Lebewesen derselben Art. Im Laufe des Lebens dieser Individuen sind diese durch ihre Gene besser oder schlechter an ihr Umfeld angepasst. Daraus entsteht die individuelle natürliche Fitness, welche bestimmt wie wahrscheinlich die Weitergabe der Gene an die nächste Generation ist. In der Natur stellt sich das konkret dar, indem zum Beispiel der besser an sein Umfeld angepasste Löwe besser Jagen kann. Dadurch steigt die Wahrscheinlichkeit, dass sich der Löwe fortpflanzen kann. [1]

Für die Benutzung in der Informatik wurde dieses Konzept algorithmisch nachgeahmt. Gelöst werden Probleme, bei denen ein variierbarer Lösungsansatz bekannt ist und eine optimale Lösung gesucht wird. Dabei werden die verschiedenen Variationsmöglichkeiten der Lösungen, die sogenannten Gene, binär dargestellt. So setzen sich die Eigenschaften eines Individuums aus den Werten seiner Gene zusammen. Der Algorithmus beginnt in der **Initialisierung**. Dort sollte eine breite Auswahl an möglichen Lösungen getestet werden. Für diesen Test wird die **Fitness-Funktion** benötigt, welche festlegt, wie gut ein bestimmtes Individuum der Population diese Aufgabe erfüllt hat. Danach entscheidet der **Selektionstyp** anhand des Fitnesswertes eines jeden Individuums, ob es „überlebt“ und ob seine Eigenschaften in die nächste Generation übergehen. Daraufhin entsteht durch **Rekombination** selektierter Individuen eine neue Generation mit derselben Populationsgröße, wie die vorherige. Diese neue Generation verändert sich dann nochmals durch **Mutation**. Dafür wird vorher ein Mutationsgrad festgelegt, welcher je nach Höhe des Grades eine Mutation der Gene gewisser Individuen vornehmen kann. [9]

3 Bestandteile des Frameworks und wie diese zusammenkommen

Aus Kapitel 2.3 ist jetzt bekannt, welche Anforderungen an einen evolutionären Algorithmus gestellt werden, um die optimale Lösung eines Problems zu finden. Im Folgenden soll dieses Gerüst auf die eingangs besprochene Problematik, einer effizienten Geometrie für ein „Insert“, angewendet werden. In Kapitel 3.1 wird beschrieben, wie ein Test für eine bestimmte Geometrie aussieht. Das Kapitel 3.2 beschreibt, wie die Phasen des Algorithmus aussehen, also wann und wie getestet wird. Und in Kapitel 3.3 wird gezeigt, wie diese Überlegungen aus Kapitel 3.2 in ein konkretes und implementierbares Framework verpackt werden.

3.1 Die Finite Element Analyse

Für das Testen einer Geometrie muss ihr Verhalten bei Einwirken von Last ausgewertet werden. Dafür wird der jeweilige Einsatz in einer Platte unter Kraftausübung simuliert.

3.1.1 Das Modell

Der hier betrachtete Einsatz hat auf Grundlage der Optimierungen von Johansson,2008 [7] die Form einer Sanduhr. Allerdings befindet sich inmitten der Sanduhr eine Stahlschraube, inklusive einer Hülse, in der diese sitzt. Eingebettet ist der Einsatz in eine quadratische Platte aus zwei Kohlefaserschichten, welche jeweils eine Seitenlänge von 148mm hat und aus 0°/90° verwobenen Kohlefasern und einer Schaum Füllung besteht. Für die Darstellung im Finite Element Modell wird nur ein Viertel der Platte inklusive Einsatz modelliert, da diese symmetrisch zur X- und Y-Achse ist (siehe Abb. 4). So kann Rechenzeit gespart werden, ohne das Ergebnis dabei zu verfälschen. Auf die Schraube wird im Laufe des Tests eine Kraft in Z-Richtung ausgeübt, währenddessen die gesamte Platte durch eine Fixierung im Umkreis von 125 mm um den Einsatz herum festgehalten wird. Dieser Testaufbau ist analog zu dem Pullouttest, welcher ebenfalls in J. Schwennen, 2016 [6] genutzt wird, auf dem die folgende Arbeit basiert. Die Kraft, welche auf die Schraube einwirkt, wird als "gleichmäßig", also linear ansteigend gewählt, um etwaige Schwingungen zu verhindern. Die Feinheit der Zerteilung der Objekte für das Modell wird zugunsten der Laufzeit einer Optimierung eher gering gewählt. Um sicherzustellen das durch diese geringe Feinheit keine, von der Realität abweichenden Ergebnisse entstehen, wurden einzelne Tests ein zweites Mal mit einem feineren Netz gemacht. Es zeigte sich, dass die entstehenden Werte um maximal 10% abweichen.

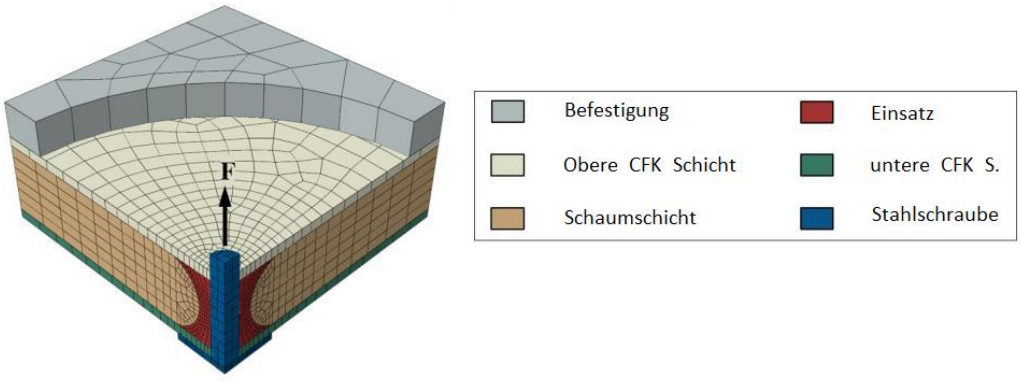


Abbildung 4 Finite Element Modell der Testplatte mit Einsatz

3.1.2 Parametrische Geometrie

Für eine effektive Optimierung mittels eines evolutionären Algorithmus muss man die Geometrie des Einsatzes auf ein paar grundlegende Werte herunterbrechen, damit der Algorithmus mit diesen Werten als Gene arbeiten kann. Dafür wird zunächst aus dem dreidimensionalen Körper eine zweidimensionale Form gemacht (siehe Abb. 5). Diese 2D-Form ist der Querschnitt des Einsatzes, der durch Rotieren entlang der Z-Achse wieder zu der Annäherung des Einsatzes wird. Um diese 2D-Geometrie noch weiter zu vereinfachen, wird sie durch zwei Dreiecke und ein Rechteck angenähert. Diese Formen werden widerum durch fünf Parameter beschrieben: R_0, R_T, R_B, H_B und H_T. R_0 gibt die Stärke des Rechtecks wieder, welches die Stahlhülle der Schraube komplett ummantelt. R_T gibt den Radius des Einsatzes am oberen Ende wieder, während R_B den unteren Radius beschreibt. H_T und H_B beschreiben die Höhen des oberen und unteren Dreiecks. Da die Schraube, um die sich die Einsatzform wickelt, einen eigenen Radius von 4mm hat, muss der geringste Radius des Einsatzes größer als 4mm sein. Der maximale Radius eines Einsatzes ist bei 34mm begenzt, damit die Geometrie nicht zu groß wird. Deshalb können die Werte R_0, R_T und R_B zwischen 4 mm und 34 mm variieren. Die Höhe der gesamten Platte ist 20 mm, deshalb müssen die Höhen H_B und H_T addiert immer 20 mm ergeben. Innerhalb dieses Rahmens können die Werte jeweils zwischen 1 mm und 20 mm variieren.

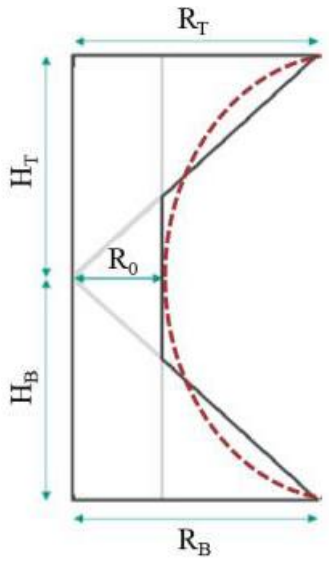


Abbildung 5

3.1.3 Modellgenerierung

Um das Abschneiden einer jeder getesteten Geometrie bewerten zu können, muss für jede einzelne Geometrie eine Finite Element Analyse (FEA) durchgeführt werden. Während dieser Analyse muss jeweils ein FEM generiert werden. Damit die Masse an Tests bewältigt werden kann, muss also die Modellgenerierung automatisiert werden. Für das Ausführen der FEA wurde das Abaqus Scripting Interface benutzt. Ohne weitere Hilfe durch Automatisierung muss jedes Modell über eine graphische Oberfläche eines CAD Programms erstellt werden. Da Abaqus allerdings Python unterstützt, ist es möglich das FEM komplett automatisiert auf Textbasis zu erstellen. Damit dies funktionieren kann, müssen alle zu testenden Geometrien in eine Datei hinterlegt werden, sodass das Abaqus-Phyton-Skript diese einlesen kann. Aus diesen Daten des 2D-Modells erstellt das Skript dann durch rotieren einen 3D-Körper, welchen es in den Rest des FEM einsetzt. Neben den Objekten an sich werden auch viele andere Randwerte automatisiert durch Algorithmen von Abaqus festgelegt, wie zum Beispiel: Feinheit der Elemente, Materialien, etc. Die Ergebnisse eines jeden Test bzw. einer jeden FEA werden in eine eigene Abaqus Modell Database (.cae) geschrieben. Wie diese Ergebnisse aussehen, wird in Kapitel 3.2.3 genauer beschrieben.

3.1.4 Die Ergebnisse

Das Ergebnis eines Tests soll die Effizienz einer Geometrie eines Einsatzes bzw. eines Sets an Parametern bewerten. Damit die große Anzahl an ausgeführten Tests in einer akzeptablen Laufzeit ausgewertet werden kann, wird das Ergebnis einer jeden FEA auf zwei Werte reduziert. Der erste Wert ist festgelegt als maximale Kraft („maximum load“). Diese beschreibt die im Laufe des Tests größte auf die Schraube eingewirkte Kraft, bevor die Platte oder der Einsatz derart nachgibt, dass keine größere Kraft mehr darauf ausgeübt werden kann. Der zweite Wert heißt „first failure“ (F(x)). Dieser beschreibt das erste Nachgeben des Materials. F(x) soll beschreiben, wann der Schaumkern oder eine Kohlenstofffaser zum ersten Mal reißt, bricht oder auf sonstige Art nachgibt. Dies wird durch ein Absinken der Reaktionskraft um 1%, im Vergleich zur eintreffenden Aktionskraft erkannt. Wenn dieses Nachgeben eine Abweichung von 1% oder mehr erzeugt, wird davon ausgegangen, dass eine Verletzung des Materials stattgefunden hat und diese Verletzung die Kraft aufgenommen hat. Bei allen Veränderungen der Reaktionskraft unter 1% bzw. unter 100N wird von Schwingungen im Material ausgegangen. V(x) beschreibt das Volumen des Einsatzes.

3.2 Evolutionäre Algorithmen angewendet

Im folgenden Abschnitt soll gezeigt werden wie der Algorithmus die Testergebnisse aus Kapitel 3.1 bewertet, verändert und schlussendlich eine optimale Geometrie findet.

3.2.1 Implementierung

Für eine Erweiterung des Phyton Skripts für einen evolutionären Algorithmus wird eine open source Bibliothek namens Galileo eingebunden [3].Die Bibliothek stellt zwei Klassen zur Verfügung, die für das Arbeiten des Algorithmus implementiert werden müssen: Population und Chromosom. Ein Chromosom steht für ein Individuum einer Generation. In unserer Anwendung ist ein Individuum eine mögliche Geometrie eines Einsatzes, welche wir testen möchten. So definieren sich die Eigenschaften eines jeden Individuums durch ein Set an Geometrieparametern und dem Fitnesswert. Eine Population repräsentiert eine Generation, also eine Menge von Individuen. Der Algorithmus arbeitet auf einer Population und geht folgende Schritte durch: Testen (Kapitel 3.2), Selektieren (Kapitel 3.2.3), Rekombination (Kapitel 3.2.4), Mutation (Kapitel 3.2.5) und Ersetzen. Also wird, wie in der biologischen Evolutionstheorie, jede Generation getestet. Je nach Fitness der Individuen und dem Vorliegen von Mutation, entsteht eine neue Generation, welche die alte ersetzt.

3.2.2 Fitness Funktion

Die Fitness Funktion soll die Effizienz der einzelnen Individuen bewerten. Dafür bricht es die FEA Ergebnisse auf einen Wert (f(x)) herunter. Die Eigenschaften des evolutionären Algorithmus sorgen daraufhin dafür, dass die Individuen künftiger Generationen diesen Wert besser erfüllen bzw. optimieren. Würde man nur „maximum load“ oder „first failure“ optimieren, würde man einen Einsatz erzeugen, welcher für viel Stabilität sorgt. Allerdings würde die Geometrie des Einsatzes nur in die maximale Größe wachsen, was zum Ansteigen des Gewichts des Einsatzes führt, sodass dieser zu schwer für die gewünschte Anwendung wäre. Außerdem zeigt sich durch den Test in J. Schwennen, 2016 [6], dass „maximum load“ und „first failure“ oftmals korrelieren. Dies hat zur Bedeutung, dass es reicht einen der Werte zu optimieren. Aus diesem Grund soll die gewählte Funktion (f(x)) das „first failure“ (F(x)) maximieren und das Volumen (V(x)) möglichst gering halten (siehe Abb. 6). Damit später priorisiert werden kann, welchen Wert die Fitness-Funktion eher optimieren soll (F(x) oder V(x)), werden zunächst beide Werte auf eine Zahl zwischen 0.0 und 1.0 normiert. Dafür werden die Werte jeweils durch ihr größtes Vorkommen geteilt. Danach bekommen beide Werte eine Variable zugeordnet: W_F für normiertes F(x) und W_V für normiertes V(x). Allerdings wird vorher Eins minus der normierte V(x)-Wert gerechnet, dadurch wird f(x) je größer, desto kleiner V(x) ist. Da der Algorithmus ein besonders großes f(x) sucht, muss V(x) dafür möglichst gering sein. W_V und W_F sollen addiert 100 ergeben, sodass die genaue Prozent Verteilung zwischen W_V und W_F der Priorisierung der einzelnen Werte entspricht.

$$f(x) = W_F \times \frac{F(x)}{20.1kN} \times W_V \times (1 - \frac{V(X)}{72cm^3})$$

Abbildung 6 die benutzte Fitness-Funktion

3.2.3 Selektion

Durch den jedem Individuum zugewiesenen Fitnesswert f(x) können die Individuen direkt miteinander verglichen werden. Um aus diesen Werten jene Individuen auszuwählen, welche durch Rekombination ihre Gene in die nächste Generation einbringen dürfen, werden drei Auswahl- / Selektionsmethoden in das Framework implementiert: Fitness-Proportionale-Auswahl, Rang-Selektion und Elite-Rang-Selektion. Bei der Fitness-Proportionale-Auswahl wird der individuelle Fitnesswert eines jeden Einsatzes durch den in der Generation höchsten Fitnesswert geteilt. Der dadurch erhaltene relative Fitnesswert sagt aus, wie gut das Individuum im Vergleich zum besten Individuum abgeschnitten hat. Nach diesem Wert kann dann geurteilt werden, wer für die Rekombination ausgewählt wird. Für die anderen beiden Auswahlmethoden werden die Individuen absteigend nach ihrem Fitnesswert geordnet. Daraufhin wird anhand ihrer Ränge entschieden. Für Rang-Selektion wird häufig eine größere Anzahl an Individuen gewählt als bei Elite-Rang-Selektion. Die Elite-Rang-Selektion sorgt dadurch, dass nur die besten Individuen die nächste Generation erzeugen, dafür, dass die Unterschiedlichkeit der Individuen schneller gegen einheitliche Generationen konvergiert. Der Nachteil dieser Methode ist, dass andere Arten von Lösungen weniger Zeit haben sich zu entwickeln. Dies kann dazu führen, dass eine bessere Lösung verloren geht. Ein Beispiel für diesen Effekt wäre ein

Vergleich des Fortbewegungstempo von Menschen und Nilpferden. Selektiert man dieses Szenario von Anfang an sehr stark, vergleichbar mit der Vorgehensweise der Elite-Rang-Selektion, so würden nur Nilpferde überleben, da sie mit ihrem vierbeinigen Gang schneller als der Urmensch sind. Verringert man allerdings den Selektionsdruck, so kann der Mensch den aufrechten Gang perfektionieren und wird dadurch schneller als ein Nilpferd. Diesen Effekt der Auswahlverzerrung kann man mit einer schwächeren Rang-Selektion entgegenwirken. Hier steigt die Anzahl der benötigten Generationen, bevor eine eindeutige Konvergenz auftritt. Je nachdem, wie der Anteil der Individuen in der Fitness-Proportionale-Auswahl gewählt wird, kann dieser Effekt auftreten. Da das hier entstandene Framework breit aufgestellt sein soll, werden alle drei Methoden implementiert. Ziel ist es, bei zukünftigen Auswertungen von Geometrien, den Selektionsdruck durch die verschiedenen Methoden variieren zu können. Auch eine Kombination der Methoden ist möglich. Nach mehreren Versuchen zeigt sich, dass die Elite-Rang-Selektion die beste Möglichkeit für das gewählte Muster ist. Sie reduziert die Laufzeit stark und führt trotzdem zu optimalen Ergebnissen. Der Auswahlverzerrungseffekt kann auch hier auftreten. Allerdings kann dem in diesem Fall durch mehrmaliges Ausführen des Algorithmus entgegengewirkt werden, denn dann scheint der Algorithmus immer wieder aus einer zufälligen Grundmenge auf dieselbe, wahrscheinlich optimale, Lösung zu kommen.

3.2.4 Rekombination

Im Rekombinationsschritt werden aus der Menge der selektierten Individuen jeweils zwei Eltern Individuen gewählt, um einen Nachkommen zu erstellen. Dies geschieht solange, bis die Anzahl der Nachkommen der gewünschten Größe der nächsten Generation entspricht. In dem Framework werden zwei Rekombinationsmethoden implementiert: uniform crossover und flat crossover. Bei uniform crossover werden für jedes Gen des Nachkommens zufällig ein Gen einer der beiden Elternteile gewählt, so dass jedes Gen eine 50% Chance besitzt weitergereicht zu werden. Bei flat crossover wird einen Nachkommen erzeugt, der eine lineare Kombination der beiden Eltern ist. Für den hier besprochenen Fall wurden beide crossover-Methoden genutzt.

3.2.5 Mutation

Nachdem die Rekombination eine neue Generation an Nachkommen erzeugt hat, soll diese die alte Generation ersetzen. Um die genetische Vielfalt weiter zu verstärken und der evolutionäre Algorithmus auch andere, eventuell bessere, Kombinationen erkunden kann, wird die Mutation eingesetzt. Die Mutation soll die Gene um einen vorher gewählten Wert, die Mutationsrate, verändern. Dafür werden wieder zwei Varianten gewählt: Einerseits die zufällige Mutation, welche jedes Gen eines jeden Individuum zufällig um einen Wert innerhalb der gewählten Mutationsrate verändert. Andererseits die uniforme Mutation, welche einen weiteren Parameter, die Mutationsbreite, nutzt. Mit diesem Parameter kann man die Anzahl der mutierenden Individuen verringern. Die uniforme Mutation sollte angewendet werden, wenn eine Veränderung eines der Gene

sehr starken Einfluss auf die Fitness des Individuums hat. Für den hier besprochenen Fall zeigte sich, dass uniforme Mutation mit 30% Mutationsbreite zu einer besseren Laufzeit führte, ohne die Ergebnisse stark zu verändern. Da jedoch akzeptable Laufzeiten bereits erreicht wurden, konnte zufällige Mutation verwendet werden. Damit einhergehend wächst die Breite an möglichen Ergebnissen.

3.3 Das Framework

Das in Kapitel 3.2 beschriebenen Vorgehen soll nun in ein konkretes Gerüst in Phyton implementiert werden, damit bei Ausführen des Skripts eine optimale Geometrie gefunden werden kann.

3.3.1 Die Struktur

Das Framework besteht aus drei Modulen: Simulation, Hantel und Misc. Das Simulations-Modul besteht aus zwei Klassen: Job und Manager (siehe Abb. 7). Der Manager enthält immer ein Objekt der Populationsklasse (Kapitel 3.1.3), welches die aktuelle Generation repräsentiert. Die Individuen / Chromosomen (Kapitel 3.1.3) dieser Population werden dann jeweils der Job-Klasse übergeben. Diese führt dann die FEA und die Auswertung für das Parameter-Set aus. Diese Daten werden dann an das Managerobjekt zurückgegeben. Das Managerobjekt führt dann die Selektion, Rekombination und Mutation der Generationen aus, bis eine eindeutige Konvergenz der Individuen zu erkennen ist. Das bedeutet, dass sich im Laufe der Generationen eine Art der Geometrie durchsetzt und in der Population ausbreitet. Diese ist dann die gefundene optimale Lösung und wird an den Nutzer zurückgegeben. Das Hantel-Modul differenziert zwischen Definitionen und Funktionen, die zum FEM gehören, diese werden zum Beispiel für die analytische Volumen Berechnung benötigt. Auf dem Misc-Modul befinden sich alle weiteren Programme, wie zum Beispiel die Skripte zum Löschen von alten Daten.

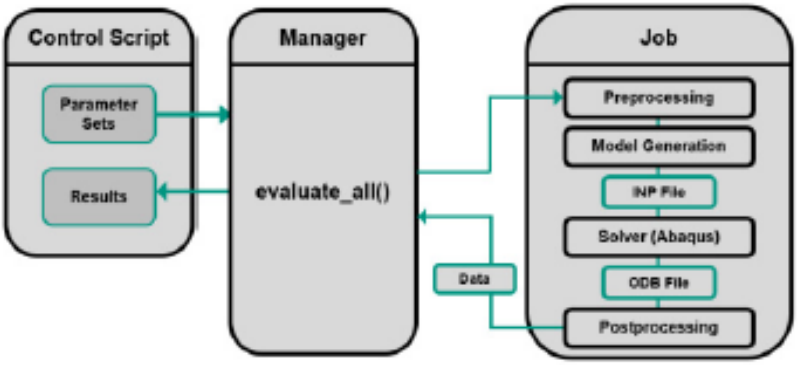


Abbildung 7 Framework des Evolutionären Algorithmus

3.3.2 Parallelisierung

Für die Parallelisierung der Tests gibt es den Vorteil, dass die Tests einer Generation nicht voneinander abhängen. Das ist der Fall, da ein Test nur von den Eingabeparametern abhängt und daraus Ausgabeparameter erstellt. Dieses Verhalten eignet sich sehr gut für eine Parallelisierung der Tests / Jobs in einer Generation. Im Framework ist der Manager zuständig für die Parallelisierung. Dieser bekommt die Parametersets der ersten Generation übergeben und erstellt daraus eine Liste an auszuführenden Job-Objekten. Beim Aufrufen der Manager-Funktion evaluate_all() werden dann alle Job-Objekte ausgeführt. Dabei kann der Manager die einzelnen FEA parallelisieren, da sie nicht voneinander abhängig sind. Daraus folgt, dass die Laufzeit des Algorithmus direkt von der verfügbaren Rechenleistung, sprich freien CPUs, abhängt.

Um die benötigte Laufzeit der Test weiter zu verringern, werden die bereits getesteten Parameter Sets und ihre Ergebnisse gespeichert, sodass für doppelt vorkommende Sets nicht erneut getestet werden muss. Außerdem werden die Zeitschritte, während der Auswertung des FEM, also die Sekunden Schritte, in denen jeweils die Verteilung der Kräfte im Modell betrachtet werden, verlängert, sodass das Modell weniger häufig berechnet werden muss. Das führt zu einer Laufzeitverkürzung, aber wird in der FEA durch eine Erhöhung der Dichte der Objekte erreicht. Das kann, bei einer zu großen Wahl der Dichte, zu einer Abweichung der Ergebnisse führen. Ein Abweichen der Ergebnisse würde sich durch eine größere Menge an kinetischer Energie, also unüblich viel Bewegung der Elemente, während der FEA zeigen. Um die entstehenden Ergebnisse möglichst akkurat zu halten, wird die Menge der entstehenden kinetischen Energie während des Versuches aufsummiert, daraufhin wird diese Summe mit Summen aus anderen Versuchen verglichen.

4 Zusammenfassung / Ergebnis

Insgesamt wurden in mehreren Ausführungen des Algorithmus über 65.000 individuelle Parametersets analysiert. Die Abbildung 8 zeigt diese Ergebnisse geplottet auf einer Skala zwischen „first failure“ des Einsatzes (invers skaliert) in kN und Gewicht in g. Je näher sich die Ergebnisse am Ursprung, der Ecke unten links, befinden, desto besser schnitt die Geometrie des Einsatzes ab, da dort die Masse geringer wird, aber der „first failure“ steigt. Die Querlinien, welche von oben links nach unten rechts gehen, beschreiben die Fitnesswerte der Geometrien. Wie oben beschrieben, sieht man, dass die Fitness in Richtung der unteren linken Ecke zunehmend ist. Durch das Plotten der Ergebnisse, ist eine klare Pareto-Front erkennbar. Das bedeutet, dass eine Grenze zu erkennen ist, an

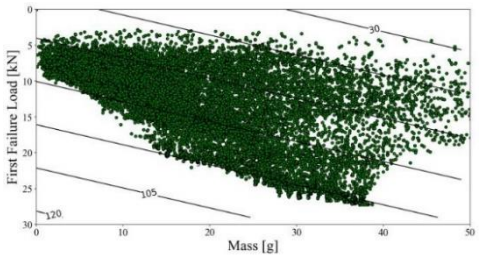
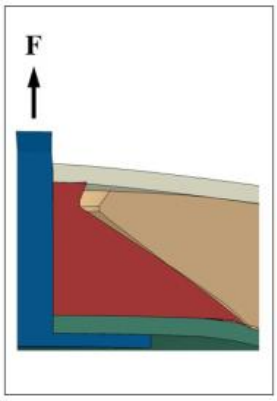


Abbildung 8

der es eine Menge an optimalen Lösungen gibt, diese liegt diesem Fall ca. bei einem Fitnesswert von 90. Dies sind die Individuen, gegen die die Lösungen des evolutionären Algorithmus konvergieren und welche als optimale Ergebnisse ausgegeben werden.

Wie in Kapitel 3.2.2 beschrieben, soll die Anwendung mithilfe des Algorithmus eine Geometrie finden, welche nicht nur „maximum load“ oder „first failure“ optimiert, da die daraus entstehenden Geometrien zu schwer sind, sondern auch das Volumen des Einsatzes betrachten. Nach mehreren Testläufen zeigt sich, dass f(x) bei einer Gewichtung von W_V = 65% und W_F = 35%, also einer Priorisierung von geringem Volumen über der von spätem „first failure“, effiziente Lösungen für den hier beschriebenen Kontext erzeugt. Die daraus entstehende Geometrie ist in Abbildung 9 abgebildet. Zu sehen ist, dass der Einsatz eine abfallende, Trichter ähnliche, Fläche bildet, welche die Kraft gleichmäßig in die Platte verteilt. Vergleicht man die Reaktionskräfte der optimierten Geometrie im Vergleich zur ursprünglichen Geometrie, erkennt man ein deutlich späteren „first failure“. Wie in Abbildung 10 zu sehen, gibt die optimierte Geometrie das erste Mal bei 20,4kN nach, währenddessen die ursprüngliche Geometrie schon bei 8,32kN nachgab. Allerdings erkennt man auch, dass die „maximum load“ der beiden Geometrien kaum unterschiedlich ist, da das Verhalten bei einer Lasteinleitung optimiert wurde, war damit zu rechnen.



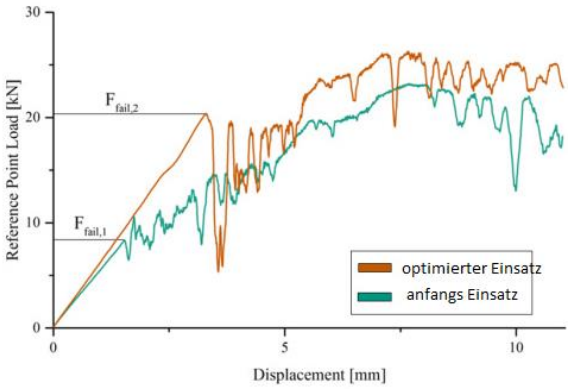


Abbildung 9 Finale Geometrie

Abbildung 10 Vergleich der optimierten Geometrie zur anfangs Geometrie

Aus den vorliegenden Ergebnissen zeigt sich, dass sich das Framework mit Hilfe eines evolutionären Algorithmus, wie gewünscht, für die Optimierung von Einsätzen zur Lasteinleitung in Faser-Kunststoff-Verbund Materialien eignet. Wenn vorher die Randbedingungen, wie welcher FKV Werkstoff genutzt wird, wo der Einsatz sitzt etc., auf den jeweiligen Fall anpasst werden, kann das in Python auf Basis von Abaqus und Galileo implementierte Framework automatisiert eine optimale Geometrie eines Einsatzes erarbeiten. Nach mehrfacher Ausführung des Algorithmus zeigte sich, dass die Laufzeit variieren kann. Dies hängt hauptsächlich von zufällig erstellten Zahlen zu Beginn der ersten Generation und der Mutation ab. Wodurch die Anzahl der Generationen bis es zu einer Konvergenz kommt variieren kann. Des Weiteren kann durch die Auswahl der verschiedenen Selektions-, Rekombinations- und Mutationsmethoden maßgeblich auf die Laufzeit und die Anzahl der erkundeten

Individuen Einfluss genommen werden. Durch diese Möglichkeiten und den hohen Grad an Parallelisierung konnten Laufzeiten von unter 12 Stunden erreicht werden.

In der Zukunft sollten weitere Möglichkeiten den Algorithmus zu variieren implementiert werden. Außerdem sollte das Framework auch für nicht symmetrische Geometrien erweitert werden.

5 Literaturverzeichnis

[1] C. Darwin (1986). *Die Entstehung der Arten durch natürliche Zuchtwahl / Übersetzt von Gerhard Heberer.* Ditzingen: Reclam.

[2] G. W. Ehrenstein (2006). *Faserverbund-Kunststoffe.* ISBN 3-446-22716-4: Hanser.

[3] D. Goodman-Wilson. (2017-02-19). *a Distributed Genetic Algorithm.* https://github.com/DEGoodmanWilson/Galileo.

[4] H. Mainka, O. Täger, E. Körner u.A. (2015). *Lignin – an alternative precursor for sustainable and cost-effective automotive carbon fiber.* https://www.sciencedirect.com/science/article/pii/S2238785415000599: SienceDirekt.

[5] J. Schwennen, L. Kalbhenn, J. Klipfel u. A. (2017). Evolutionary Optimization of the Failure Behavior of Load Introduction Elements Integrated During FRP Sandwich Structure Manufacturing

[6] J. Schwennen, V. Sessner, J. Fleischer (2016). A New Approach on Integrating Joining Inserts for Composite Sandwich Structures with Foam Cores. *6th CIRP Conference on Assembly Technologies and Systems (CATS),*, S. 310-315.

[7] H. Johansson (2008). THE OPTIMISED POST-FITTED FOAM SANDWICH INSERT. *13th. European Conference on Composite Materials* , S. 1-7.

[8] U. Thalhofer, M. Mayr (1993). Numerische Lösungsverfahren in der Praxis. *ISBN 3-446-17061-8*, S. 312.

[9] T. Weise (2009). *Global Optimization Algorithms.*

6 Abbildungsverzeichnis