DLR-IB-FA-BS-2019-43

Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs

Bachelorarbeit

Peter Löwen

Deutsches Zentrum für Luft- und Raumfahrt

MH-FA-56-FB03_v1.3

Institut für Faserverbundleichtbau und Adaptronik

DLR-IB-FA-BS-2019-43

Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs

Zugänglichkeit: Allgemein zugänglich

Braunschweig, *05, 2019*

Abteilungsleiter: Dr.-Ing. Tobias Wille

Der Bericht umfasst: 74 Seiten

Autoren: *Peter Löwen*

Autor 2 / Betreuer: Falk Heinecke

Deutsches Zentrum DLR für Luft- und Raumfahrt

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Peter Löwen

Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Peter Löwen

Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Flugzeugbau
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Institut für Faserverbundleichtbau und Adaptronik
Lilienthalplatz 7
38108 Braunschweig

Erstprüfer: Prof. Dr.- Ing. Eckart Nast
Zweitprüfer: Dipl.- Ing. Falk Heinecke

Abgabedatum: 03.12.2018

Zusammenfassung

Name des Studierenden
Peter Löwen

Thema der Bachelorthesis
Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs.

Stichworte
Faserverbundwerkstoffe, Unsicherheiten, Virtual Allowables, Virtuelles Testen, Sensitivitätsanalysen

Kurzzusammenfassung
In dieser Arbeit geht es um die Unsicherheiten von Faserverbundstrukturen, welche während der Herstellung und Fertigung auftreten. Diese werden zusammengefasst und ihre Einflussgrößen mit zwei unterschiedlichen Sensitivitätsanalysen berechnet. Hierzu werden Versuchspläne erstellt und mit Hilfe Finiten-Elemente-Methode simuliert (virtuelles Testen). Darauf aufbauend werden die effektiven Kennwerte für Steifigkeiten und Festigkeiten von Faserverbundstrukturen mit Open Source Programmen bestimmt (Virtual Allowables).

Name of Student
Peter Loewen

Title of the paper
Coping with manufacturing uncertainties in the analysis and evaluation of composite structures in the context of overall aircraft design.

Keywords
Composite, uncertainties, virtual allowables, virtual testing, sensitivities analysis

Abstract
This thesis deals with the uncertainties of fiber composite structures that occur during manufacturing. These are summarized and their influencing variables are calculated with two different sensitivity analyses. Experimental designs are created and simulated by finite element methods (virtual testing). On this basis, the effective characteristic values for stiffness and strength of fiber composite structures are determined with open source programs (virtual allowables).

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

FVW	Faserverbundwerkstoffe
AFP	Automated fiber placement
UD	Unidirektional
FEM	Finite-Elemente-Methode
VA	Virtual Allowables
E-Modul	Elastizitätsmodul
FE	Finite Elemente
DMS	Dehnungsmessstreifen
FO	First order
SO	Second order
TO	Total order

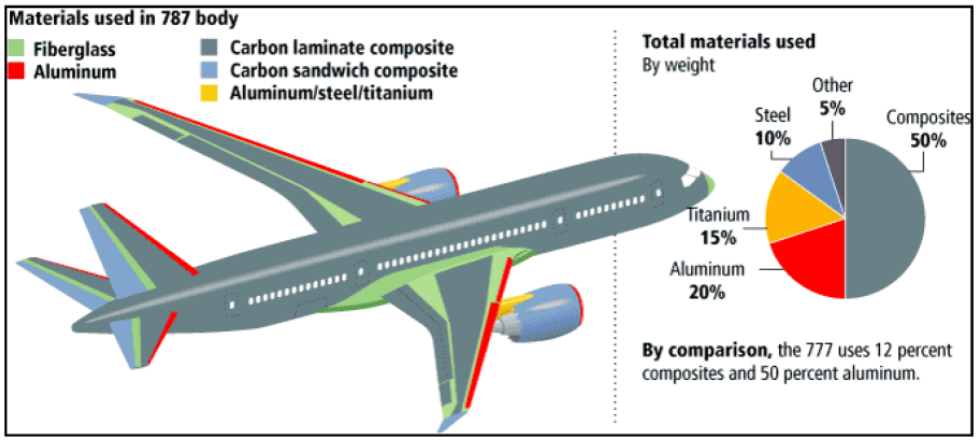
Symbolverzeichnis

Symbol	Bezeichnungen	Einheit
E	Elastizitätsmodul	N/mm^2 bzw. MPa
G	Schubmodul	N/mm^2 bzw. MPa
ν	Querkontraktionszahl	-
φ	Faservolumengehalt	%
A	Flächeninhalt	mm^2
V	Volumen	mm^3
F	Kraft	N
σ	Normalspannung	N/mm^2 bzw. MPa
ε	Dehnung	-
Δl	Verschiebung	mm
b	Breite	mm
l	Länge	mm
h	Höhe	mm
Δb	Breitenänderung	mm
τ	Schubspannung	N/mm^2 bzw. MPa
K	Korrekturfaktor	-
μ	Mittelwert	-
σ	Standardabweichung	-

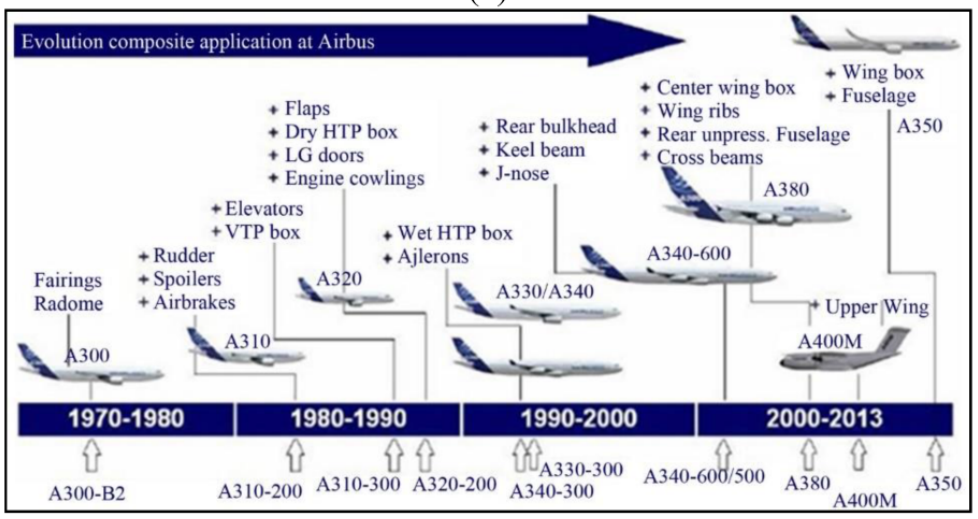
1 Einleitung

Die vorliegende Arbeit beschäftigt sich mit dem Umgang von fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen. Das Ziel ist es, die Unsicherheiten exakter einordnen und bewerten zu können, so dass durch Simulation eine genauere Bestimmung und Auslegung des Faserverbundwerkstoffes (FVW) erfolgen kann. Auf diese Weise könnte Gewicht und damit Kosten eingespart werden. Diese Arbeit wurde mit Salome Meca und Code_Aster durchgeführt, um gleichzeitig herauszufinden, ob diese Open-Source-Programme eine mögliche Alternative zu den zurzeit verwendeten kostenpflichtigen Programmen darstellen.

FVW finden sich aufgrund ihrer hohen Festigkeits- und Steifigkeitswerte sowie ihres geringeren Gewichts immer häufiger im Leichtbau wieder. Insbesondere im Flugzeugbau finden Faserverbunde immer mehr Anwendung, da hier das Gewicht eine erhebliche Rolle spielt (Abbildung 1).



(a)



(b)

Abbildung 1: Faserverbunde im Flugzeugbau bei Boeing und Airbus (Sante)

Dem gegenüber steht eine große Anzahl an Unsicherheitsfaktoren, welche durch hohe Sicherheitsfaktoren kompensiert werden. Dies hat zur Folge, dass die Fertigungs- und Berechnungskosten hoch ausfallen und die Bauteile überdimensioniert werden.

FVW sind eine Kombination aus zwei Materialien (Abbildung 2): aus Fasern und der sie umgebenden Matrix. Die Faser dient im Wesentlichen der Festigkeit und Steifigkeit der Struktur. Elementare Aufgaben der Matrix sind die Formgebung, die Druckstabilität und der Schutz der Faser. Die Berechnung der Faserverbunde wird durch die meist anisotropen Eigenschaften erschwert (→ die Eigenschaften an einem Punkt sind richtungsabhängig).

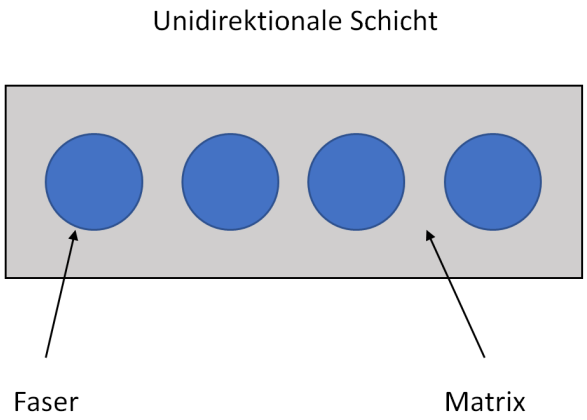


Abbildung 2: Darstellung einer Unidirektionalen Schicht

Bei großen Bauteilabmaßen, hoher Stückzahl und genauerer Fertigung werden automatisierte Faserablege-Maschinen (AFP) verwendet. Hierbei werden die einzelnen Faserverbunde in einzelnen unidirektionalen (UD) Schichten zu einem Bauteil gelegt. Dabei entstehen Fertigungsabweichungen, welche die Steifigkeiten und Festigkeiten des Bauteils verändern. Ein Beispiel dafür wären Versetzungen der einzelnen UD-Schichten. Diese Abweichungen müssen mit den entsprechenden Sicherheitsfaktoren in der Berechnung berücksichtigt werden.

Die Faserverbunde werden mit einer dreißigprozentigen Sicherheit der ultimativen Lasttragfähigkeit ausgelegt (Frank Abdi, 2009). Daher sind genauere Betrachtungen und Berechnungen der Unsicherheiten und deren Auswirkung sinnvoll, damit im Anschluss durch eine bessere Abschätzung die Bauteile noch leichter werden können. Um dies zu erreichen, werden die Unsicherheiten mit den jeweiligen Auswirkungen zusammengetragen und mit der Finite-Elemente-Methode (FEM) berechnet.

Die mit den Berechnungen zusammenhängende numerische Simulation der modellhaften Bauteile soll die Notwendigkeit der kostspieligen praktischen Versuche reduzieren und somit Kosten einsparen. Zudem dient dies der Vorhersage, mit welcher Wahrscheinlichkeit die Bauteile versagen würden. So kann eine bessere Einschätzung zur Auslegung eines Modells getroffen werden. Das angestrebte Resultat der Arbeit soll eine virtuelle Testmethode sein, die der

Bestimmung effektiver Kennwerte für Festigkeit und Steifigkeit, sogenannter Virtual Allowables (VA) ermöglicht. VA sind die simulierten Materialkennwerte eines Materials, die Computer basiert berechnet werden. Es werden freizugängliche Opensource-Programme verwendet, um hierdurch den Kostenfaktor, der sonst mit der Berechnung entsteht, zu senken.

2 Grundlagen und Stand der Forschung

In diesem Kapitel werden die bereits zur Verfügung stehenden Ressourcen und Möglichkeiten dargelegt, auf welchen die vorliegende Arbeit aufbaut.

2.1 Grundlagen der Theorie

Zunächst werden die theoretischen Grundlagen, die eine Basis für diese Arbeit bilden, vorgestellt.

Für Laminate existieren zahlreiche Berechnungsmodelle, aber alle verfolgen dasselbe Ziel: Die Ermittlung der Antwort des Materials auf äußere physikalische Belastungen.

Die unterschiedlichen Ansätze der in der Fachliteratur existierenden Berechnungsmodelle von FVW variieren in ihrer Komplexität. Eine der einfacheren und etablierten Modelle ist die lineare Mischungsregel nach Altenbach (Holm Altenbach, 1996).

Hier wird näherungsweise angenommen, dass das Materialverhalten in der tangentialen Ebene und in der quer zur Faser liegenden Ebene ein Isotropes (→ die Eigenschaften in einem Punkt sind in allen Richtungen gleich) Verhalten aufweist. Somit kann vom Transversalen Isotropen Materialverhalten gesprochen werden (Holm Altenbach, 1996).

Eine faserverstärkte unidirektionale Einzelschicht mit einem transversal isotropen Materialverhalten ist mit folgenden fünf unabhängigen elastischen Kennwerten gekennzeichnet:

- Elastizitätsmodul (E-Modul) in Faserrichtung E_1
- E-Modul quer zur Faserrichtung E_2
- Querkontraktionszahlen ν_{12}und ν_{23}
- Schubmodul G_{12}

Das Modell der linearen Mischungsregel nach Altenbach wird nachfolgend tiefergehend erklärt. In dieser hier angewandten Berechnung werden einige Parameter, wie der thermische Einfluss und der Feuchtigkeitseinfluss, vernachlässigt. Die Vereinfachungen dienen in diesem Fall der besseren Bestimmung des expliziten Flächeninhaltes der Faser (A_F) und der Matrix (A_M). Der Flächeninhalt wird zur Berechnung des gesamten Volumens des Laminats (V_L) benötigt, welches sich wiederum aus dem Volumen der Matrix (V_M) und der Faser (V_F) zusammensetzt. Das Volumen wird wiederum zur Berechnung des Faservolumengehalts (φ) benötigt, welches für die Berechnung des E-Moduls in Faserrichtung (E_1) und quer zur Faserrichtung (E_2) benötigt wird. Der Faservolumengehalts (φ) setzt sich wie folgt zusammen:

$$\varphi = \frac{A_F}{A_F + A_M} = \frac{V_F}{V_F + V_M} \quad (1)$$

Zur Bestimmung des E-Moduls in Faserrichtung (E_1) wird die lineare Mischungsregel aufgestellt und angewandt. Dies erfolgt mit der Hilfe des hookeschen Gesetzes und den im Laminat auftretenden Kräften. Das hookesche Gesetz besagt, dass die Spannung in Faserrichtung (σ_{L1}) das Produkt von Längsdehnung (ε_{L1}) und dem E-Modul in Faserrichtung (E_1) ist, bzw. die Kraft (F) dividiert durch die Fläche (A).

$$\sigma_{L1} = \varepsilon_{L1} * E_1 = \frac{F}{A} \quad (2)$$

Mit der Längsdehnung ε_{L1}:

$$\varepsilon_{L1} = \frac{Längenänderung\ \Delta l}{Gesamtlänge\ l} \quad (3)$$

Die Kräfte im Laminat (F_{L1}) setzen sich aus der Summe der Kräfte in der Matrix (F_{M1}) und in der Faser (F_{F1}) zusammen:

$$F_{L1} = F_M + F_F \quad (4)$$

Der Zusammenhang zwischen den auftretenden Kräften (F_{L1}), den Querschnittsflächen (A) und den Spannungen in Faserrichtung (σ_{L1}) ergibt sich wie folgt:

$$F_{L1} = \sigma_{LM} * A_M + \sigma_{LF} * A_F \quad (5)$$

$$\sigma_{L1} * A_L = \sigma_{M1} * A_M + \sigma_{F1} * A_F \quad (6)$$

Mit dem Faservolumenanteil (φ) kann die Spannung nun vereinfacht folgend ausgedrückt werden:

$$\sigma_{L1} = \sigma_{M1} * (1 - \varphi) + \sigma_{F1} * \varphi \quad (7)$$

Werden nun die beiden Gleichungen (2) und (7) ineinander eingesetzt, so erhält man:

$$\varepsilon_{L1} * E_1 = \varepsilon_M * E_M(1-\varphi) + E_F * \varepsilon_F * \varphi \tag{8}$$

Vereinfacht man diese Gleichung, ergibt sich:

$$E_1 = E_M(1-\varphi) + E_F * \varphi \tag{9}$$

Um bei der Berechnung die Faserwelligkeit mit zu berücksichtigen, wird ein Korrekturfaktor (K_1) eingeführt. Für die Berechnung des E-Moduls in Faserrichtung mit dem Korrekturfaktor lautet die Formel anschließend:

$$E_1 = K_1 * E_F * \varphi * \left(1 + \frac{1-\varphi}{\varphi} * \frac{E_M}{E_F}\right) \tag{10}$$

Zur Bestimmung des E-Moduls quer zur Fasserrichtung (E_2) wird ebenso die lineare Mischungsregel angewandt. Hierbei ist zu beachten, dass die Beanspruchung der Faser und der Matrix in einer Reihenschaltung erfolgt. Somit gilt in diesen Schichten:

$$F_{L2} = F_M = F_F \tag{11}$$

$$\sigma_{L2} = \sigma_{M2} = \sigma_{F2} \tag{12}$$

Und es ergibt sich die Gesamtdehnung (ε_{L2}) von:

$$\varepsilon_{L2} = \frac{Breitenänderung\ \Delta b}{Gesamtbreite\ b} \tag{13}$$

Werden die Formeln (12), (13) und das hookesche Gesetz (2) zusammengefasst, ergibt sich das E-Modul quer zur Faserrichtung (E_2):

$$E_2 = \frac{E_F * E_M}{(1-\varphi) * E_{M+}\varphi * E_M} \tag{14}$$

Die typischen Versagensarten von FVW sind: Faserbeulen, Faserbrechen, Matrixrissbildung oder Delamination. Diese werden üblicherweise nach zwei Kriterien beurteilt: nach dem Maximalspannungskriterium oder dem Maximaldehnungskriterium (Holm Altenbach, 1996). Diese Kriterien sind einfach zu bestimmen, haben aber den Nachteil, dass sie nur das Versagen der einzelnen Schicht liefern, nicht jedoch das Versagen des ganzen Bauteils. Ebenso sind diese Kriterien nicht anwendbar auf bereits beschädigte Bauteile.

Die Festigkeiten in der UD-Schicht werden experimentell gemessen. Dabei gibt es unterschiedliche Kennwerte für Zug- und Druckfestigkeit. Bei einer Druckbelastung wird von Druckfestigkeit gesprochen, wobei die relevanten Druckfestigkeiten in Faserrichtung (R_{11c}), quer zur Faserrichtung (R_{22c}) und längs zur Faserrichtung (R_{33c}) gekennzeichnet sind. Bei einer Zugbelastung sind die Zugfestigkeiten in Faserrichtung (R_{11t}), die Zugfestigkeit quer zur Faserrichtung (R_{22t}) und längs zur Faserrichtung (R_{33t}) gekennzeichnet. Die drei möglichen Schubrichtungen sind (R_{12}), (R_{13}) und (R_{23}).

Bei dem Maximalspannungskriterium wird von Versagen gesprochen, sobald mindestens eine Spannungskomponente den zugehörigen Festigkeitsgrenzwert erreicht. Wie in Abbildung 3 veranschaulicht, ist die Belastungsrichtung ein wichtiger Aspekt der Betrachtung. Ist die Belastung eine Zugkraft ($\sigma_{L1} > 0$) in Faserrichtung, so wird bei Versagen der Komponente von Faserbruch gesprochen, wohingegen bei einer Druckkraft ($\sigma_1 < 0$) in Faserrichtung von Mikroknicken gesprochen wird. Ist die Beanspruchung quer zur Faserrichtung ($\sigma_{L2} > 0$), handelt es sich um Zwischenfaserbruch.

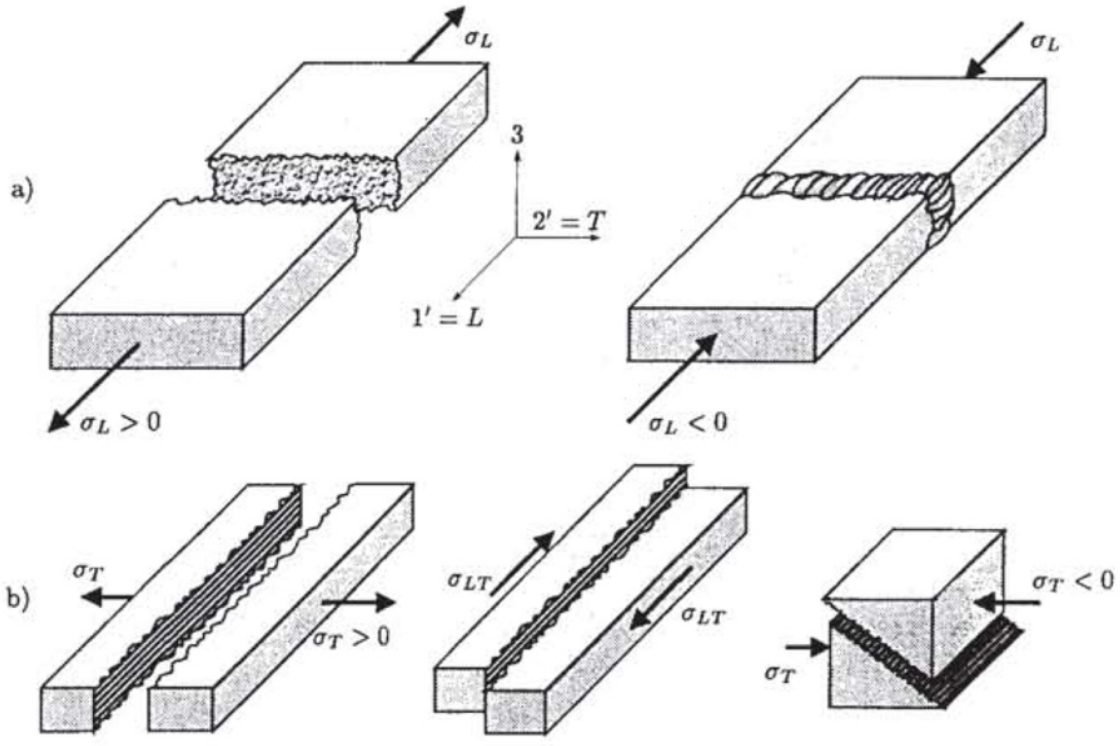


Abbildung 3: Maximalspannungskriterium (Holm Altenbach, 1996)

Eine weitere Versagensart, die zur Auslegung herangezogen wird, ist "First-Ply-Failure". Diese tritt ein, wenn bei einem Laminat eine einzelne Schicht versagt. In Folge wird das ganze Bauteil als defekt eingestuft, obwohl es noch Kräfte aufnehmen könnte und nicht komplett versagt hat.

Um die Spannungen in den Einzelschichten zu bestimmen, muss eine Transformation der globalen Spannungen in lokale Spannungen erfolgen. Dies geschieht mithilfe der Transformationsgleichung. Es wird auf eine Herleitung verzichtet und auf die Ausführungen von Altenbach (Holm Altenbach, 1996) verwiesen:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -2\sin\theta\cos\theta \\ -\sin\theta\cos\theta & \sin\theta\cos\theta & (\cos^2\theta - \sin^2\theta) \end{bmatrix} * \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \quad (15)$$

Die verwendeten Materialien und deren Herstellungsverfahren müssen für die Luft- und Raumfahrt mittels strenger und zahlreicher Tests hinsichtlich ihrer Zulässigkeit qualifiziert werden. Das Ziel dieser Tests besteht darin, Prozesse, Methoden und die Spezifikationen zu validieren. Abbildung 4 zeigt einen Bausteinansatz, der die Zertifizierung und den Produktionsqualifizierungsablauf darstellt. Hieran wird deutlich, dass sich die Anzahl der durchgeführten Tests zum Sockel hin vervielfacht. Es werden viele Tests in der Komponenten-Ebene durchgeführt, da sich die eventuell späteren Änderungskosten um ein Vielfaches in der Struktur- und Subkomponenten-Ebene verteuern.

Zur Veranschaulichung wird ein Beispiel betrachtet. Das Leitwerk einer Boeing 777 wurde über 8000 Mal getestet. Davon dienten alleine 2000 Versuche zur Ermittlung der Steifigkeit des Laminates (Sang Yoon Park, 2017). Hierbei sind die sogenannten A-Basis- und B-Basis-Werte zu bestimmen. Diese Werte sind statische Werte, die zur Auslegung herangezogen werden. Die B-Basis ist die Bewertung des zehnten Perzentils einer Stärkeverteilung mit 95 % Vertrauensniveau, und die A-Basis ist die Bewertung des ersten Perzentils einer Festigkeitsverteilung mit 95 % Vertrauensstufe.

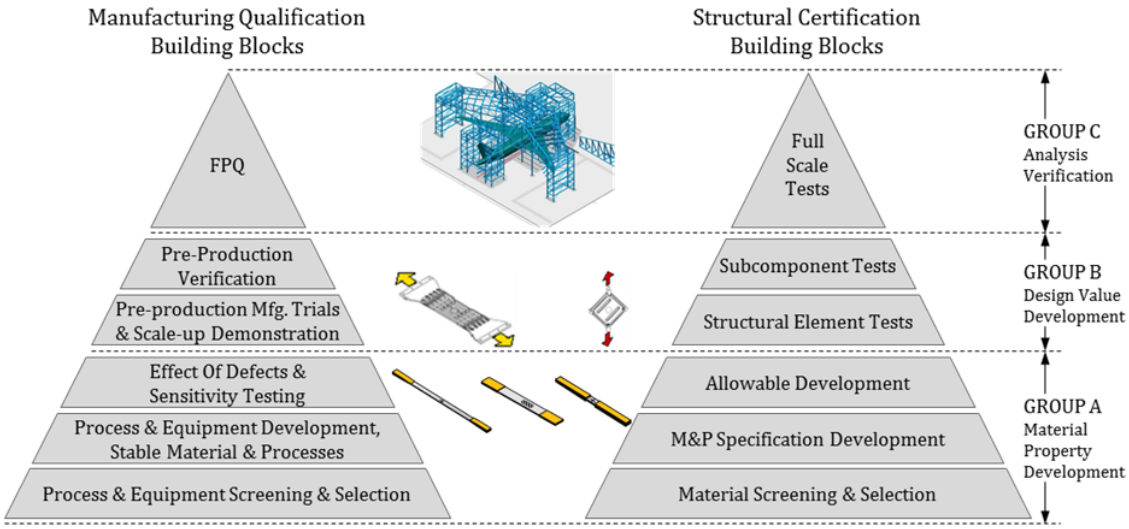


Abbildung 4: Bausteinansatz nach MIL-HDBK-17F, zeigt die strukturelle Zertifizierung und Produktionsqualifizierung (Young, 2013; U.S. Department of Transportation, 2002)

Um eine gleichbleibende Materialqualität zu gewährleisten, wurde ein Prozess geschaffen, der den Zertifizierungsvalidierungsprozess von Materialien und Prozessspezifikationen miteinander verknüpft (Abbildung 5). In diesem Prozess werden die Materialeigenschaften festgelegt und anschließend überprüft, ob die jeweilige Spezifikation eingehalten wird.

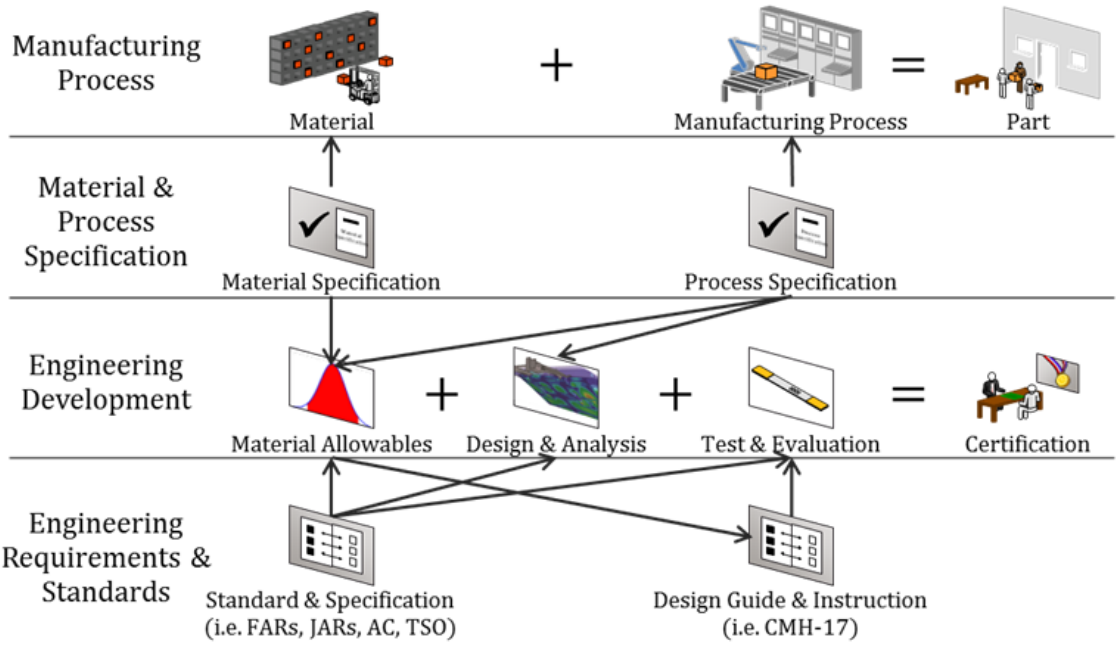


Abbildung 5: Zertifizierungsvalidierungsprozess von Material und Prozessspezifikation (Sang Yoon Park, 2017)

Die Spezifikation nach Airbus AITM1-0008 definiert eine Methode, nach der die Festigkeiten von Faserverbunden bei einer Druckprüfung zu ermitteln sind. Hierbei

werden die Geometrien der Testproben unter Einhaltung entsprechender Normen vorgegeben.

Für die Berechnungen werden folgende Formeln in der Spezifikation bereitgestellt:

Die Druckspannungsformel (σ_{cu}) mit maximaler Last (F_u), Breite der Probe (b) und Laminatdicke (t_n):

$$\sigma_{cu} = \frac{F_u}{t_n * b} \ (MPa) \tag{16}$$

Der Kompressionselastizitätsmodul (E_c), welcher von der Spannungsdehnungskurve, wie in der Abbildung 6 zu sehen, abgeleitet wird:

$$E_c = \frac{\Delta F}{b * t_n * \Delta\varepsilon_x} \ (MPa) \tag{17}$$

Hierbei ist ΔF die Erhöhung der Last und $Av(Pu)$ der Durchschnitt der Versagenslast von dem Set der getesteten Proben.

$$\Delta F = \frac{Av(Fu)}{2} - \frac{Av(Fu)}{10} \tag{18}$$

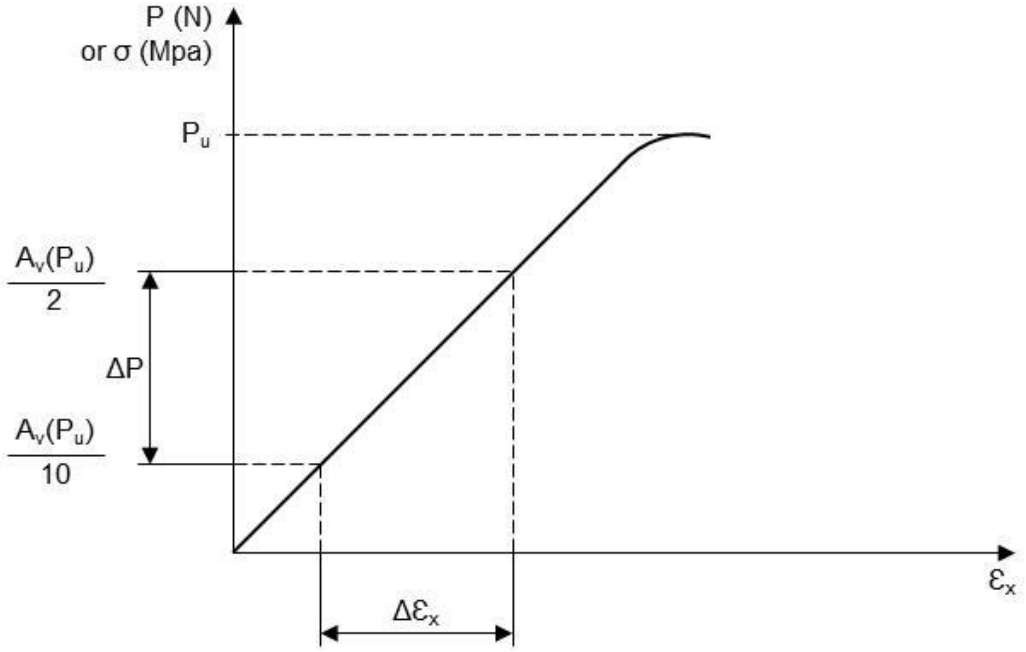


Abbildung 6: Spannungs-Dehnungskurve oder Last-Dehnungskurve

Die Testproben sind in zwei Kategorien unterteilt: "THICK" und "THIN". Die in "THIN" eingeteilten Proben haben normalerweise eine Höhe von ca. 2 mm. Die in "THICK" eingeteilte Proben haben hingegen eine Höhe von ca. 4 mm, 8 mm oder auch 12 mm.

Die Proben werden zudem in vier verschiedene Geometrietypen, von A1 bis A4, unterteilt. Die erforderliche Mindestpresskraft (rote Pfeile in Abbildung 7 und Abbildung 8), welche zwischen 100 kN und 1000 kN liegt, hängt von der Dicke des jeweiligen Typs ab.

Alle für den Test verwendeten Messeinrichtungen müssen kalibriert sein. Die Pressmaschine sollte innerhalb der geforderten Presskraft von 10 % bis 100 % die folgenden Kriterien erfüllen:

- Die Maschine muss die Scherkräfte übertragen können (Abbildung 7).
- Die Maschine muss die kombinierten Kräfte übertragen können (Abbildung 8).
- Die Maschine muss die aufgebrachte Last messen und die Verschiebungen anzeigen können.

Weitere Kriterien und der Aufbau der Maschine können in der Spezifikation nachgeschlagen werden (AIRBUS, März 2015).

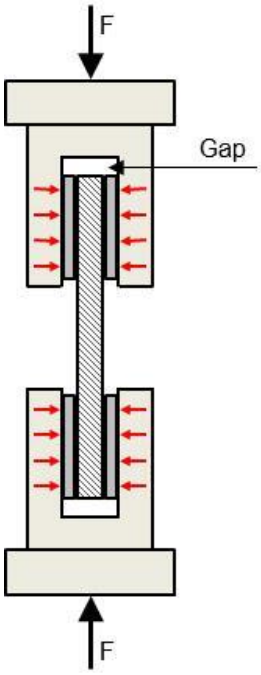


Abbildung 7: Druckkraft – Scherbelastung (AIRBUS, März 2015)

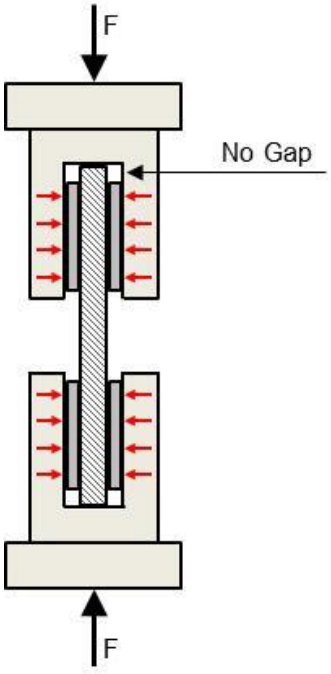


Abbildung 8: Druckkraft - kombinierte Belastung (AIRBUS, März 2015)

Um vergleichbare Ergebnisse zu erhalten, müssen mindestens sechs Proben (entsprechend Abbildung 9) für jede Test-Konfiguration getestet werden. Für das Testen der Testpaneele der Typen A1 bis A4 sind "Tabs" auf die Paneele aufzubringen, welche die Scherlast und die Temperatur möglichst ohne Energieverlust auf die Paneele übertragen. Die weiteren Anforderungen an die Tabs sind aus der Spezifikation zu entnehmen (AIRBUS, März 2015).

Bei jeder Probe muss die Höhe mit einer Bügelmessschraube gemessen werden, wobei eine Toleranz von $\pm$0,01 mm zulässig ist. Die Höhe und die Länge der Probe werden mit einem Messschieber und einer Toleranz von $\pm$0,1 mm gemessen.

Die Dehnung der Proben wird mit einem Dehnungsmesstreifen (DMS) gemessen. Die DMS werden auf jeweils einer Seite des Testpaneels, mit einer Toleranz von $\pm$2° zur 0°-Achse aufgebracht. Die Messung der beiden Dehnungen muss unabhängig erfolgen, um ein Beulen oder eine Biegung zu erfassen. Für die Geometrietypen A1 und A2 werden 8 DMS verwendet, für A3 und A4 müssen 12 DMS eingesetzt werden.

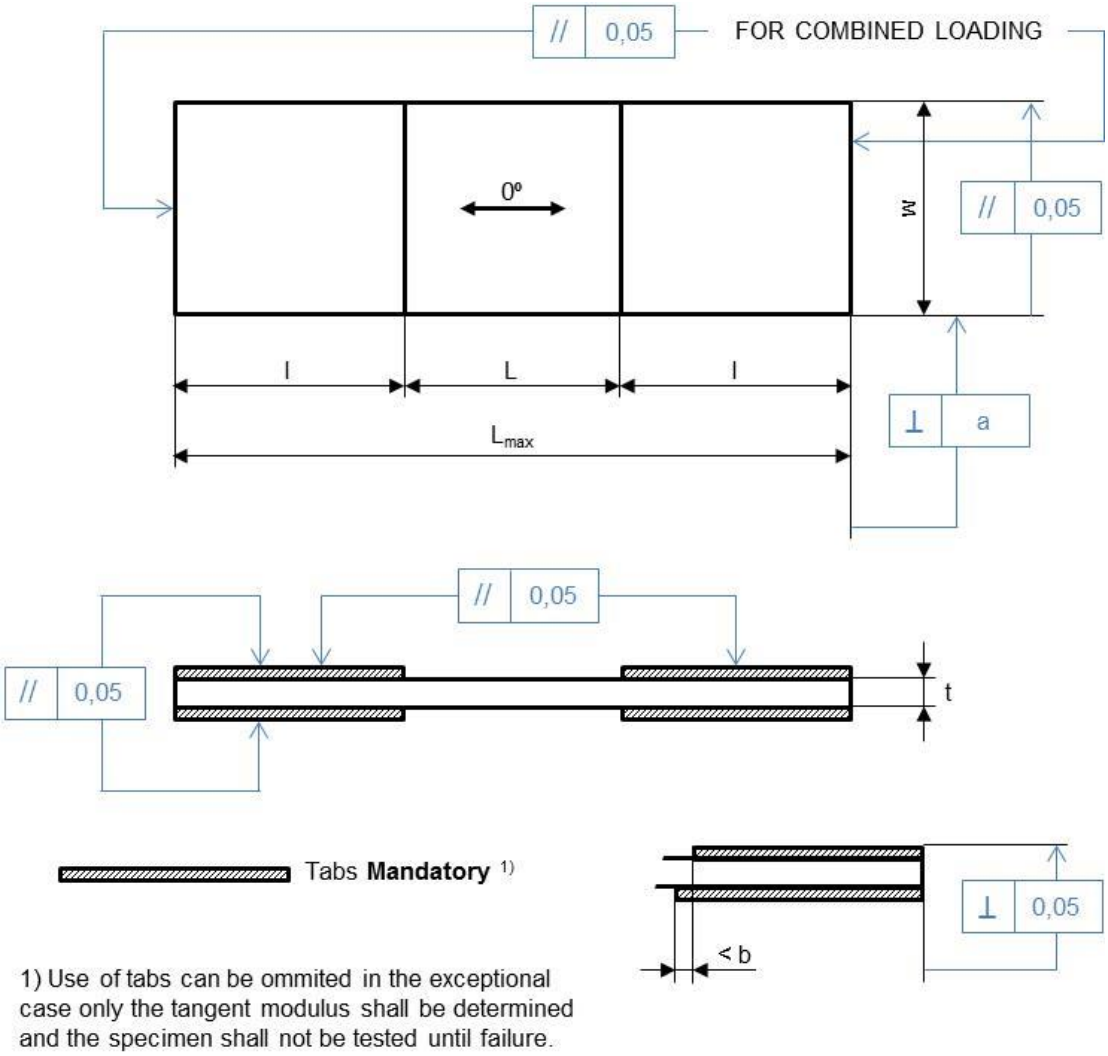


Abbildung 9: Druckprobengeometrie und „Tab-“Abmaß von AITM 1-0008

Ist die maximale Last (P_u) gemessen worden, kann daraus mit Gleichung (16) die maximale Spannung berechnet werden.

Die Testergebnisse können nur dann verwendet werden, wenn die Druckproben eine der Versagensformen aufweisen, welche in Abbildung 10 zu sehen sind.

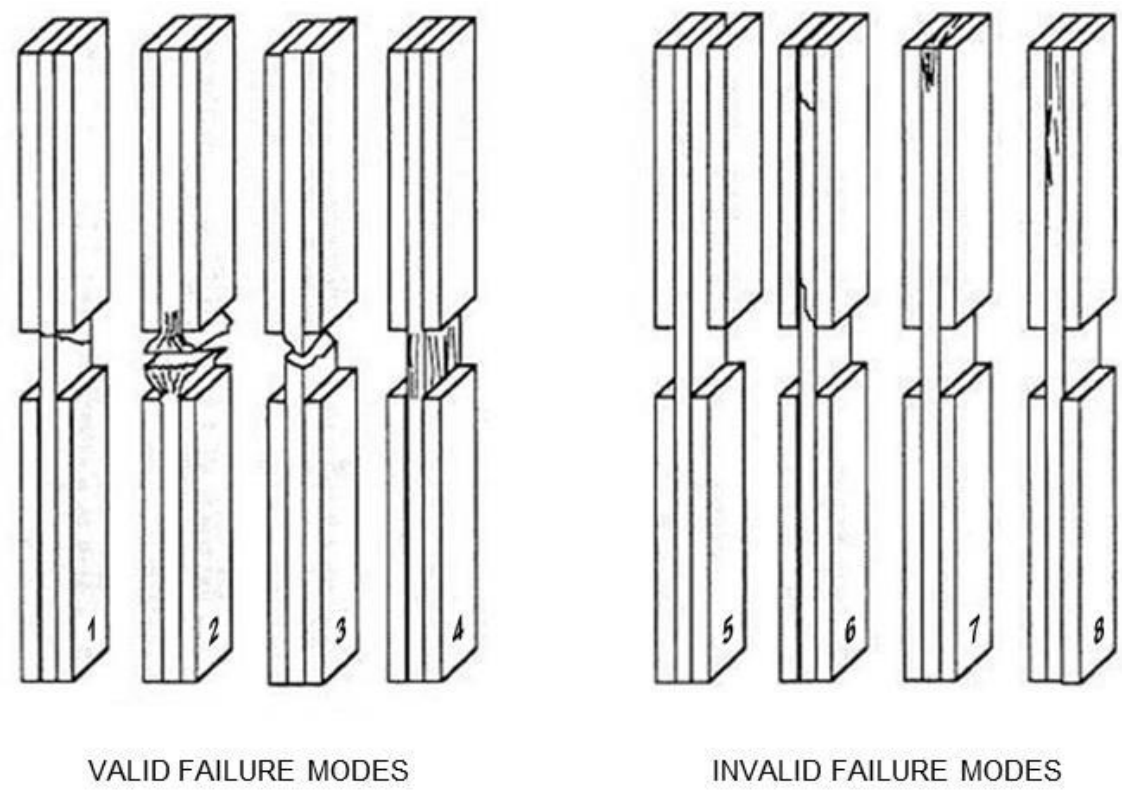


Abbildung 10: Versagensform für Testpaneele (AIRBUS, März 2015)

2.2 Stand der Forschung

Dieser Abschnitt präsentiert die unterschiedlichen Sachstände in der Forschung zu diesem Thema.

Die Vielfalt der Ansätze zur Beurteilung der Unsicherheiten und ihrer Auswirkungen entsteht durch die große Bandbreite der möglichen Varianten. Die ersten Unsicherheiten treten bei dem Herstellungsprozess von Matrix und Faser auf, beispielsweise hinsichtlich der Eigenschaften der einzelnen Faser und der Matrix. Hier können Faserwelligkeiten, Mikrorisse in den Fasern und in der Matrix, Variationen des Faservolumengehalts auftreten. Im Verarbeitungsprozess kommen die Lagenorientierung, die Stapelfolge, die Dicke des Laminats (Caracciolo, 2014), die Umgebungsbedingungen, und viele weitere Faktoren (Sang Yoon Park, 2017) dazu.

In einer Arbeit betrachtet Park (Sang Yoon Park, 2017) ebenso den Einfluss der Maschinen und Werkzeuge, die benutzt werden. Hier haben die Sauberkeit, die Toleranzen und der Zustand der Maschinen einen erheblichen Einfluss. Mesogtis (Mesogitis, 2013) hingegen untersucht den Aspekt von trockenen Textilien und Prepregs und berücksichtigt die auftretenden Unsicherheiten beim formen und drapieren während der Imprägnierung mit den verschiedenen möglichen Effekten. Die Unsicherheiten in der Herstellung und Fertigung von Faserverbunden sind oftmals vergleichbar. So können einige Punkte einfach übertragen und übernommen werden.

Für das virtuelle Testen von FVW, um die VA zu bestimmen, gibt es einen FEM-basierten Ansatz. Die Software von MSC Software Digimat VA erlaubt es dem Benutzer, Versuchstests zu simulieren (MSC, 2014). Das Ziel ist hier die Vorhersage von Festigkeitswerten der Faserverbunde und deren Versagensform. Die Software kombiniert nichtlineare Micro-Mechanik, Versagensanalyse und nichtlineare FE-Berechnung miteinander. Sie erlaubt es dem Anwender, eine komplette Testmatrix zu definieren, über welche ein Material oder mehrere Materialen ausgewählt werden können. Ebenso ist es möglich, eine Matrix manuell zu generieren, welche eine oder mehrere Lagenanzahlen hat, und die Faserrichtungen festzulegen. Testfälle wie "Open Hole", "Filled Hole" und "Plain" können definiert werden und es stehen mehrere Einstellungen für die Temperatur und Luftfeuchtigkeit zur Verfügung. Zusätzlich können Materialunsicherheiten in den Herstellungs- oder Fertigungsprozess eingebunden werden. So können Unsicherheiten von Material-Eigenschaften, des Faservolumengehalts und der Faserausrichtung eingebunden werden. Im Anschluss wird die Spannungs-Dehnungskurve berechnet, ein Versagenskriterium mit Degradationsverhalten des Materials erstellt, die maximale Spannung ausgegeben und die A-Basis- und B-Basis-Werte werden erzeugt.

2.3 Unsicherheitsfaktoren

In diesem Kapitel wird ein Überblick über die Unsicherheitsfaktoren gegeben, für die eine Abschätzung und eine Analyse über die Einflussgrößen mit den jeweiligen Auswertungen erfolgt.

Bei der Herstellung und Fertigung von unidirektionalen Faserverbunden entstehen eine Anzahl an Unsicherheiten. In der Abbildung 11 sind nur einige davon visualisiert.

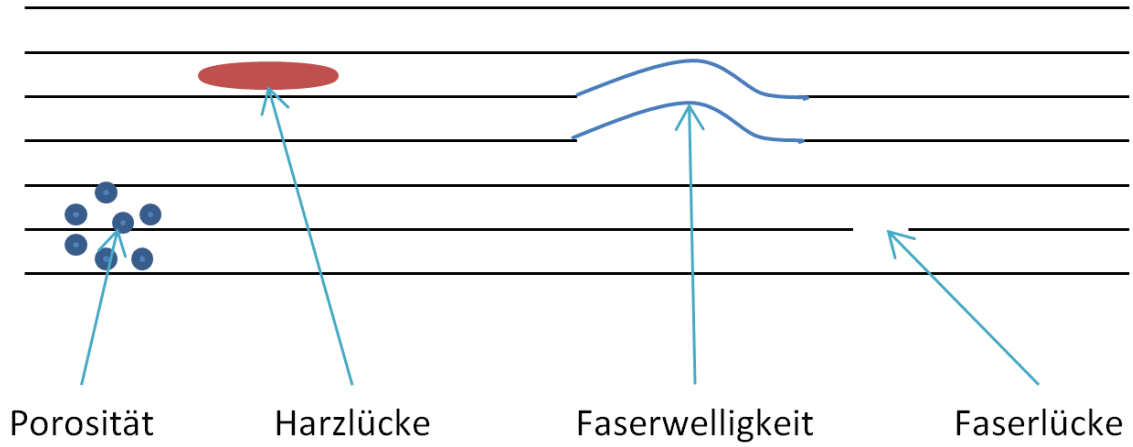


Abbildung 11: Schematische Darstellung von Fehlstellen im Laminat

Material-Parameter

Die E-Module leiten sich von der linearen Mischungsregel ab (Gleichungen (9) und (14)). Bei den Angaben der Hersteller für die Module der Faser und der Matrix ist jeweils eine Toleranz enthalten, sodass auch hier kein statischer Wert zur Beurteilung des Unsicherheitsanteils ermittelt werden kann.

Zur Veranschaulichung dieses Faktors dienen die Herstellerangaben für die Faser IM7 und die Matrix aus PETI-5. Sie schwanken von $\pm$5-10 % für die einzelnen Module (Frank Abdi, 2009). Diese Angaben stimmen auch mit der Arbeit von Chen (Xiao Chen, 2017) überein. Hier wird für die Faser T300 und die Matrix QY8911 für das E-Modul in Faserrichtung (E_1) ein Mittelwert von 1250 MPa mit der Standardabweichung von 100 MPa angegeben. Für das E-Modul quer zur Faserrichtung (E_2) ist der Mittelwert mit 300 MPa mit der Standardabweichung von 30 MPa angegeben. Auch die Abweichungen von der Querkontraktionszahl und dem Schubmodul betragen genau 10 %. Die Materialkennwerte von dem Prepreg IM7/8552 werden für die Berechnungen angenommen: E_1 = 163000 MPa, E_2 = 8500 MPa, G_{12} = 4200 MPa, ν_{12} = 0,35.

Porosität

Porosität tritt nicht nur bei der Fertigung, sondern auch schon bei der Herstellung der Matrix auf und beeinflusst so die Festigkeiten des FV (Sang Yoon Park, 2017). Während der Fertigung entsteht durch Lufteinschlüsse bei der Kollation der einzelnen Lagen Porosität (auch Harzlücke genannt). Ebenso hat der Härtungsprozess einen Einfluss, denn hierbei können sich Gase oder Feuchtigkeit im Harz freisetzen und es kann zum Aufbau von innerer Spannung kommen, welche bei der Harzhärtungsschrumpfung entsteht. Äußere Fertigungsfaktoren für Harzlücken sind zum Beispiel: schlechte Werkzeuge, eine unzureichende Lagenverfestigung, ein zu geringer Autoklaven-Druck, Vakuumverlust oder auch fehlerhaftes Material.

Ein Anteil von 1 % Porosität verringert die Festigkeiten des Faserverbundes um 5 % (Sang Yoon Park, 2017). Bei kugelförmigen Einschlüssen ist der Einfluss von Porosität auf das E-Modul als linear anzunehmen (Krimmer, 2014).

Faserwelligkeit

Faserwelligkeit kann bei der Herstellung von UD-Schichten entstehen. Sie kann sowohl in der Ebene als auch außerhalb der Ebene beobachtet werden und tritt während der Härtung durch Restspannungen innerhalb der Matrix auf.

Bei einer beispielhaften experimentellen Untersuchung der inneren Geometrie der Textilien und Prepregs (Mesogitis, 2013) mit einem dreidimensionalen- (3D) Scanner zeigte sich eine Welligkeitstoleranz von 3-4 %.

Lagenorientierung

Lagenorientierung beschreibt die Ausrichtung einer gesamten UD-Schicht. Hier entstehen Abweichungen im Produktionsprozess bei der Ablage der einzelnen UD-Schichten durch die Maschine und auch bezüglich deren Präzision.

Die Ablage einzelner UD-Schichten wird üblicherweise mit einer Toleranz von $\pm 2°$ angegeben (Krimmer, 2014). Die Abnahme des E-Moduls in Faserrichtung beträgt ~ 1 % und quer zur Faserrichtung ~ 0,1 %. Das Schubmodul nimmt um rund 0,3 % und die Querkontraktionszahl um 1,4 % zu. In einem weiteren Bericht (Sang Yoon Park, 2017) wird eine Toleranz von $\pm 3°$ angegeben.

Geometrie

Bei den Unsicherheitsfaktoren in der Geometrie handelt es sich um die gesamte Laminatlänge, die Laminatbreite und die Laminathöhe. Bei der Laminathöhe wiederum gibt es zwei Aspekte: Der eine entsteht bei der Herstellung der einzelnen UD-Schichten und der andere bei der Fertigung des gesamten Laminats. Dabei wird die Laminathöhe durch die Wärmezufuhr bei der Herstellung und Produktion sowie durch den Druck bei der Ablage der einzelnen UD-Schicht zu einem Laminat beeinflusst, denn hierbei werden die Höhenunterschiede von der UD-Schicht an die Laminathöhe weitergegeben. Um die Höhentoleranzen nominell abzubilden, werden Daten vom NCAMP-Test-Report (Research, 2011) herangezogen und ausgewertet. In diesem Bericht wurden Faserverbunde mit den

Material Hexcel 8552 IM7 auf verschiedene Arten getestet (belastet) und die gemessenen Dicken der einzelnen Proben ausgewertet. Dabei sind Dicken von 0,0059 bis 0,0078 Inch (0,14986 mm bis 0,19812 mm) gemessen worden. Somit errechnet sich eine Toleranz der einzelnen UD-Schicht von $\pm$14 %. Für die Breiten-, Längen- und Dicken-Abweichungen des Laminates werden die Toleranzen aus der Spezifikation von Airbus (AIRBUS, März 2015) verwendet. Hier ist bei einer Breite von 22 mm eine Toleranz von $\pm$0,2 mm angegeben und die Länge von 22 mm ist mit einer Toleranz von $\pm$0,5 mm versehen. Für die Variation der Laminathöhe ist eine maximale Abweichung von $\pm$10 % angegeben. Diese Toleranzen können im Python-Programm bei den Parametereingaben direkt verändert werden und bedürfen daher keiner Umrechnungen auf die zur Verfügung stehenden Parameter, die in dem Programm variiert werden können. Bei vorangegangen Arbeiten (Richard Moon, 2003) (Yves Davila, 2016) wurden die Einflüsse der Geometrie-Toleranzen von FVW geprüft, was zu dem Ergebnis führte, dass die in der Toleranz liegenden Geometrie-Abweichungen keine gravierende Auswirkung auf die Festigkeiten und Steifigkeiten von Faserverbunden besitzen. Die Geometrie-Unsicherheiten werden in den weiteren Berechnungen daher nicht weiterverfolgt.

Faserlücke

Eine Faserlücke, wie in Abbildung 11 dargestellt, ist für die Berechnungen der Unsicherheiten irrelevant, da deren Einfluss infinitesimal klein ist. Auf einer Fläche von 15 mm^2 (0,25 mm Höhe und 60 mm Breite entspricht einer Standard-UD-Schicht) gibt es bei einem Faserdurchmesser von 6 μm und einem Faservolumengehalt von ca. 60 % über 318.000 Fasern. Die Wahrscheinlichkeit, dass an genau einem Punkt so viele Fasern defekt sind, dass dies eine Auswirkung auf die Festigkeit zur Folge hätte, ist gering und wird daher im weiteren Verlauf vernachlässigt.

Faservolumengehalt

Der Faservolumengehalt ist ein wesentlicher Parameter in der Mischungsregel und beeinflusst die Module in Faserrichtung (E_1) und quer zur Faserrichtung (E_2). Bei der Herstellung der einzelnen UD-Schichten wird der Faservolumengehalt meist mit einer Toleranz von $\pm$2 % angegeben (Krimmer, 2014). Die Faservolumenschwankungen sind in dem Materialparameter enthalten und werden nicht noch einmal separat betrachtet.

Verschiebung / Maschinen

Die Herstellungs-, Fertigungs- und Prüfmaschinen haben ebenso einen Einfluss auf den FV. In der Herstellung der UD-Schicht können verunreinigte Maschinen und Änderungen im Prozessablauf die Material-Parameter beeinflussen. Diese auftretenden Unsicherheiten sind jeweils in den Toleranzen beim Material-Parameter verarbeitet. In der Fertigung von Laminaten entstehen Abweichungen wie Lagenorientierung und Porosität durch ungenaue Ablage der UD-Schicht von der Maschine und durch inkonstante Verarbeitungstemperatur. Auch diese Abweichungen sind jeweils in den Lagenorientierungen und den Materialparametern verarbeitet worden. Die Abweichungen, die durch

Prüfmaschinen entstehen, sind die Messungenauigkeiten von Messinstrumenten und die Lasteinleitung der Prüfmaschinen. Messungenauigkeiten treten bei der Benutzung der Messinstrumente auf. Diese Abweichungen, die in den Prüfmaschinen auftreten, werden mit einem Wert von $\pm$5 % angenommen.

Festigkeiten

Die Festigkeit ist ein Materialwert, der unabhängig von dem Querschnitt der Probe ist, sondern vom Werkstoff und der Art der Beanspruchung abhängt. Die Festigkeit ist die Widerstandsfähigkeit eines Werkstoffes gegen äußere Belastungen. Für den Wertebereich der Unsicherheiten liegen keine Angaben vor. Daher werden die Toleranzen der Unsicherheiten mit $\pm$5 % angenommen.

Zusammenhängende Unsicherheiten

Die Materialparameter werden durch praktische Versuche bestimmt und enthalten schon einige der zuvor aufgezählten Unsicherheiten, wie die Porosität, Faserwelligkeit und den Faservolumengehalt. Daher wird in der späteren Sensitivitätsanalyse deren Einfluss durch die Variation der Materialparameter eingebunden und bedarf keiner gesonderten Betrachtung.

Um einen besseren Überblick über die Unsicherheiten zu geben, wurden diese in Tabelle 1 zusammengefasst.

Tabelle 1: Unsicherheiten und deren Verteilungsfunktion

Unsicherheit mit Normalverteilung	**Mittelwert**	**Standard-abweichung**	**Variationskoeffizient (%)**
Material Parameter	E_1 = 163000 MPa E_2 = 8500 MPa G_{12} = 4200 MPa ν_{12} = 0,35	16300 MPa 850 MPa 420 MPa 0,035	10 10 10 10
Faserwelligkeit	0°	4°	4,44
Faservolumengehalt	59 %	2 %	3,5
Lagenorientierung	0°	3°	3,33
Lagendicke	0,25 mm	0,035 mm	14
Probengeometrie	Breite: 22 mm Länge: 22 mm Dicke: 2 mm	0,2 mm 0,5 mm 0,2 mm	0,9 2,3 10
Verschiebung	Verschiebung: 0,1 mm	0,005	5
Festigkeiten	R_{11t}: 2610 MPa R_{11c}: 1450 MPa R_{22t}: 55 MPa R_{22C}: 285 MPa R_{12}: 105 MPa R_{33t}: 64,5 MPa R_{33C}: 250 MPa R_{13}: 68 MPa R_{23}: 68 MPa	130,5 MPa 75,5 MPa 2,75 MPa 14,25 MPa 5,25 MPa 3,225 MPa 12,5 MPa 3,4 MPa 3,4 MPa	5 5 5 5 5 5 5 5 5
Unsicherheit mit Gleichverteilung	**Mittelwert**	**Abweichung**	**Variationskoeffizient (%)**
Porosität / Harzlücke	0,5 % E_1 = 163000 MPa E_2 = 8500 MPa	$\pm$ 0,5 % $\pm$ 1630 MPa $\pm$ 85 MPa	1 1 1

2.4 Vorstellung der Sensitivitätsanalysen

Um den Einfluss der Unsicherheiten auf die Ergebnisse bewerten zu können, wird eine Sensitivitätsanalyse durchgeführt. In der Studie werden die Einflüsse der einzelnen Unsicherheiten auf die Festigkeit und Steifigkeit berechnet. Es wird angenommen, dass die Unsicherheiten unabhängig voneinander sind. Diese Annahme gilt nicht für die E-Module. Weist das Material zum Beispiel ein hohes E-Modul in Faserrichtung (E_1) auf, ist mit einer sehr hohen Wahrscheinlichkeit auch ein höheres E-Modul quer zur Faserrichtung (E_2) zu erwarten. Jede Unsicherheit tritt mit einer Wahrscheinlichkeit auf, die mit einer Funktion abgebildet wird. Dabei ergeben sich zwei verschiedene Funktionsformen: Bei der Porosität sind Werte von 0 % bis 1 % zu erwarten und die Wahrscheinlichkeit, dass diese einen Wert im Bereich zwischen 0 und 1 annehmen, ist gleich. Somit wird von einer Gleichverteilung oder auch Rechteckverteilung gesprochen (Türck, 2005).

Alle weiteren Unsicherheiten aus der Tabelle 1 weisen eine Normalverteilung auf, wie in Abbildung 12 dargestellt ist. Die Werte liegen mit einer hohen Wahrscheinlichkeit in der Nähe des jeweiligen Mittelwertes. Viele Unsicherheiten lassen sich durch eine Normalverteilung beschreiben, da die Herstellungs- und Fertigungsprozesse den idealen Wert anstreben.

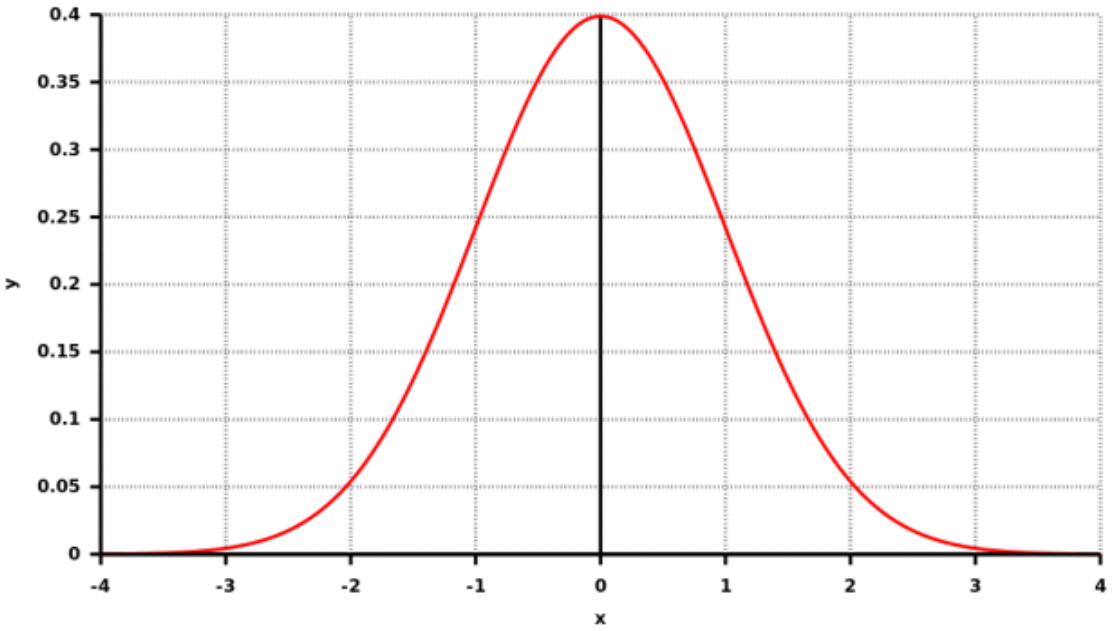


Abbildung 12: Dichtefunktion der Normalverteilung

Für eine effizientere Berechnung werden die Verteilungsfunktionen zu einer zusammengefasst. Dies geschieht nach der Copula-Theorie (Caniou, 2012). Für einen vollständigen Überblick über die Copula-Theorie kann in der Literatur (Nelson, 1999) nachgeschlagen werden.

Um die Anzahl der Unsicherheiten auf die relevanten zu reduzieren, wird ein Screening nach der Morris-Methode (Andrea Saltelli, 2004) durchgeführt. Die

Morris-Methode gibt an, welche Unsicherheiten den größten Einfluss haben und welche davon einen linearen und welche einen nichtlinearen Einfluss besitzen. Für jede Berechnung wird jeweils ein einzelner Parameter verändert und anschließend das Ergebnis ausgewertet. Dies ist für jeden eingegebenen Parameter notwendig. Um den Mittelwert bestimmen zu können, werden mindestens zwei Änderungen des jeweiligen Parameters (r) benötigt. Somit ergibt sich die Anzahl der Rechnungen (n) aus dem Produkt der Anzahl der Eingabefaktoren (k) und der Änderungsdurchläufe der Parameter (r):

$$\mathrm{n} = \mathrm{r} * \mathrm{k} \tag{19}$$

Die Ergebnisse können leicht interpretiert werden (Andrea Saltelli, 2004). Je größer das Maß des Mittelwertes (μ) ist, desto größer ist der Gesamteinfluss des Parameters auf das Ergebnis. Ein großes Maß von der Standardabweichung (σ) zeigt, dass der Eingabe-Effekt sich nichtlinear auf das Ergebnis auswirkt und/oder an Interaktionen mit anderen Parametern beteiligt ist. Für die Eingabe der Parameter müssen diese vorher normiert werden.

Um die Wechselwirkung der Parameter untereinander zu betrachten, wird die Sensitivitätsanalyse mit den Sobol-Indizes angewandt. Die Methode berechnet den direkten Einfluss der Parameter und die Wechselwirkung der Parameter untereinander auf das Ergebnis. Dabei werden alle eingegebenen Parameter gleichzeitig variiert. Dadurch sind mehrere Rechnungen notwendig, um eine genaue und verwertbare Aussage treffen zu können. Die Anzahl der Rechnungen (n) ist in der nachfolgenden Gleichung zu sehen, wobei N die Anzahl der Kombinationsmöglichkeiten zwischen den Parametern und k die Anzahl der Parameter ist:

$$n = N * (2k + 2) \tag{20}$$

Die Auswertung der Ergebnisse ist leicht zu interpretieren (Andrea Saltelli, 2004). Für jeden Parameter werden drei Ergebnisse ausgegeben: First order (FO), Second order (SO) und Total order (TO), die jeweils einen Mittelwert und eine Standardabweichung besitzen. Der Wert für die FO gibt den direkten Einfluss des Parameters auf das Ergebnis an, der Wert für die SO benennt die Wechselwirkungen der Parameter untereinander und der TO-Wert stellt die Komplexität des Funktionsverlaufes des jeweiligen Parameters dar. Das Erhöhen der Kombinationsvariation (N) minimiert die Wahrscheinlichkeit, negative Ergebnisse der Parameter, die einen geringen Einfluss auf das Ergebnis besitzen, zu erhalten. Das Auftreten negativer Parameter zeigt an, dass mehr Kombinationsvariationen (N) notwendig sind.

3 Konzept zur Bestimmung von "Virtual Allowables"

In diesem Kapitel wird ein Konzept vorgestellt, wie die "Virtual Allowables" bestimmt werden können. Es wird ein Programm geschrieben das den Ablauf der Berechnungen steuert und alle Applikationen mit einander verbindet. Die Berechnungen der Struktur werden mit Hilfe einer FEM durchgeführt.

3.1 Umsetzung

Dieser Abschnitt der Arbeit beschreibt, wie die Berechnung der VA umgesetzt wurde. Dies ist der Hauptbestandteil dieser Arbeit und erfolgt mit dem erstellten Python Programm. Für diese Berechnung sind folgende Anforderungen zu erfüllen und einzugeben:

- Eingabe der Geometrie
- Eingabe der Belastungsart
- Eingabe von Unsicherheiten und deren Verteilungsfunktionen
- Eingabe der möglichen Berechnungsarten
- Eingabe von Versagenskriterien
- Ausgabe von Spannungen in den einzelnen Schichten
- Ausgabe von Kräften in den Knoten
- Ausgabe von Festigkeiten
- Auswertung der Ergebnisse

Mit diesen erstellten Anforderungen wurde ein Konzept entwickelt, welches die VA eines FV erstellt. Dieses Konzept ist in Abbildung 13 dargestellt.

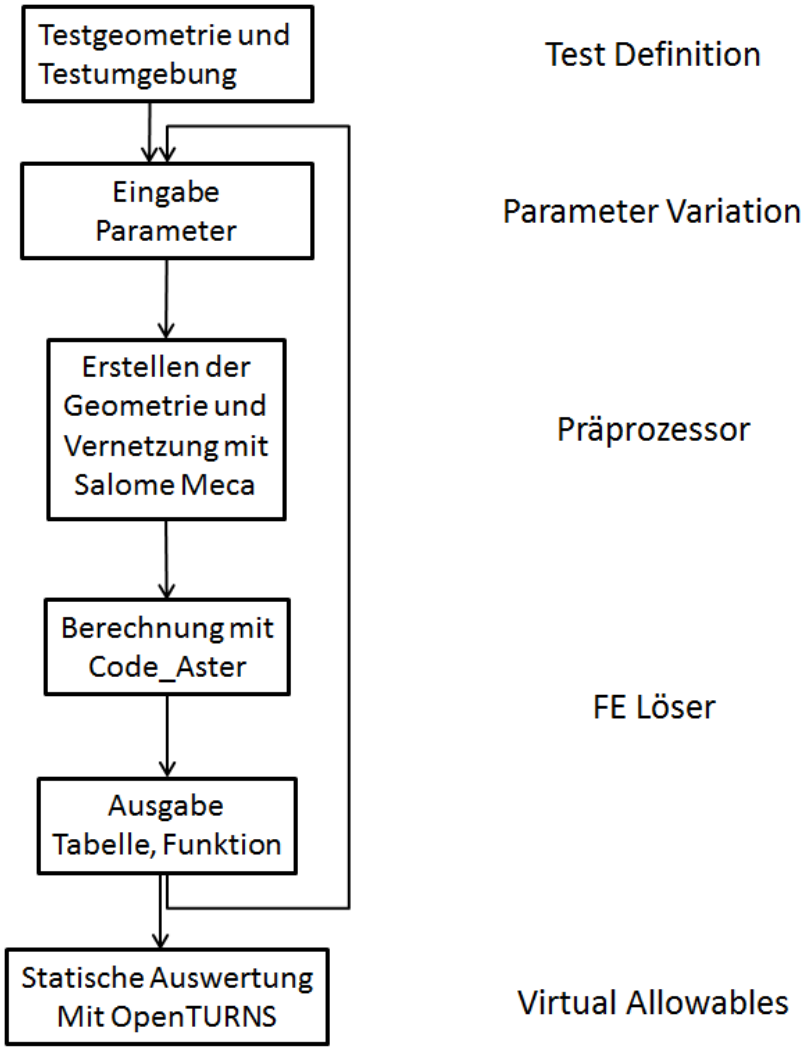


Abbildung 13: Konzept

Zu Beginn des Ablaufes erfolgt die Definition des Testes. Hier werden die grundlegenden Geometrienmaße und die Lagenanzahl eingegeben und das Testverfahren wird definiert. In diesem Fall wird eine Drückprüfung an einem ebenen Faserverbund definiert. Die notwendigen Geometriemaße werden aus der Spezifikation von Airbus AITM 0-0008 (AIRBUS, März 2015) übernommen. Bei der Druckprüfung wird der FV zusammengedrückt, bis ein Versagensfall auftritt, was in diesem Fall das First-Ply-Failure ist.

In dem zweiten Schritt erfolgt nun die Eingabe der Unsicherheiten mit den jeweiligen Verteilungsfunktionen. Dabei werden die Unsicherheiten, die in der Tabelle 1 zusammengetragen worden sind, mit ihrer Auftretenswahrscheinlichkeit errechnet und in den Prozess eingebunden. Für diese Berechnung der VA mit einer FEM muss ein Versuchsplan erstellt werden, mit dem die Sensitivitäten der Unsicherheiten berechnet werden können. Zuerst kommt die Methode nach Morris zur Anwendung, die mit kurzen Berechnungszeiten die wichtigsten Parameter herausfiltert, indem man einen ersten Eindruck des jeweiligen Einflusses der unterschiedlichen Parameter erlangt. Mit der auf diese Weise reduzierten Anzahl an Parametern wird die Sobol Indizes Methode angewandt, welche genauere

Ergebnisse liefert und die Korrelation der einzelnen Parameter untereinander berücksichtigt, dafür aber eine erhöhte Rechenzeit benötigt.

Sind alle Eingaben erfolgt, wird die Probengeometrie erstellt und anschließend vernetzt. Dieser Schritt wird mit Salome Meca (Kapitel 3.2) durchgeführt und die Geometrie wird entsprechend den zuvor erstellten Vorgaben angefertigt. Dabei wird nur der mittlere Bereich bzw. der beanspruchte Bereich, welcher für die Auswertung relevant ist, modelliert. Nach der darauffolgenden Vernetzung wird eine Datei ausgegeben, welche alle notwendigen Informationen über das Netz enthält, die für eine Berechnung mit einer FEM notwendig sind.

Für die Berechnungen der Spannungen in den Elementen und Kräfte in den Knoten wird Code_Aster (Kapitel 3.3) verwendet. Hier wird zunächst die Vernetzungsdatei eingelesen. Danach werden die Material-Parameter, die Randbedingungen, die Belastung, die lineare Berechnung und die Ergebnisdateien definiert und die Spannungen und Kräfte berechnet. Danach erfolgt die Ausgabe der Spannungen in den einzelnen Schichten und die Knotenkräfte.

Sind alle Rechnungen durchgeführt, kann die statistische Auswertung der Ergebnisse erfolgen, im vorliegenden Fall mit dem Programm OpenTURNS. Mit OpenTURNS werden ebenso die Verteilungsfunktionen der Unsicherheiten mit der Copula-Theorie zusammengefasst und die Versuchspläne erstellt, welche für die Berechnung der Sensitivitätsanalysen nach der Morris-Methode und den Sobol-Indizes geeignet sind.

Um den kompletten Ablauf zu steuern, wird ein Programm mit der Programmiersprache Python erstellt. Python ist eine weitverbreitete Programmiersprache und mit vielen Anwendungen kompatibel. Die einzelnen Programme, die für die Berechnung der VA verwendet werden, wie Salome Meca, Code_Aster und OpenTURNS, arbeiten alle mit der Programmiersprache Python.

An die Ablaufsteuerung werden folgende Anforderungen gestellt:

- Variation der Geometrieabmaße
- Variation der Lagenanzahl
- Variation der Materialparameter
- Variation der Netzverfeinerung
- Erstellen eines Versuchsplanes
- Auswertung und Ausgabe der VA

Mit diesen Anforderungen wird ein weiteres Konzept erstellt, das in Abbildung 14 veranschaulicht ist.

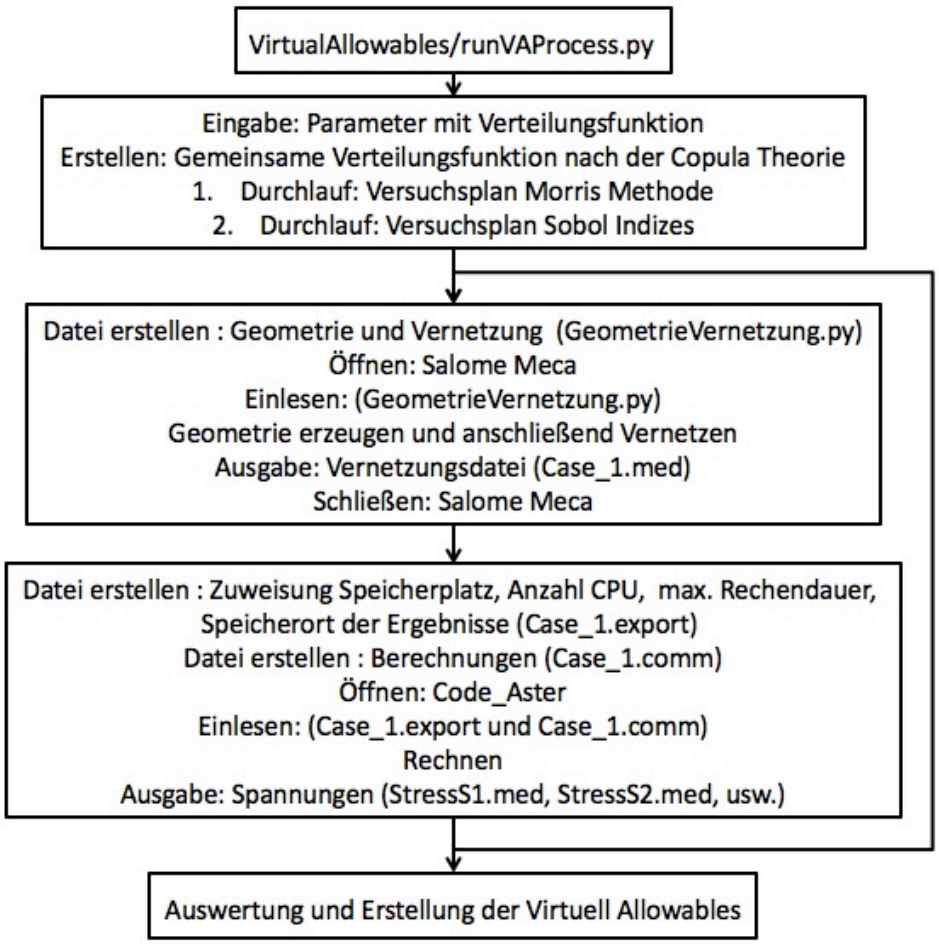


Abbildung 14: Python Konzept

Für eine vereinfachte Eingabe und Darstellung aller Parameter und die in dem Programm definierten Variablen wird ein Dictionary (Abbildung 15) angelegt. Dieses erlaubt es mittels Variablen auf deren hinterlegten Wert zuzugreifen. In diesem Dictionary sind alle Variablen mit dem zugehörigen Werte definiert (Abbildung 15). Sind diese Eingaben erfolgt, kann entschieden werden, ob die Sensitivitätsanalyse nach Morris oder mit den Sobol-Indizes durchgeführt werden soll. Die Sensitivitätsanalyse nach Morris fordert normierte Werte von null bis eins. Demzufolge werden die Parameter hierfür durch das Programm noch dementsprechend normiert. Für das Verfahren nach den Sobol-Indizes ist eine Normierung nicht erforderlich und die Werte werden ohne Bearbeitung aus dem Dictionary entnommen.

```
parametersDict = {'a':10., 'name':'Test',
                  'specimenWidth':22., 'specimenLength':22., 'tabLengthLeft':7., 'tabLengthRight':7.,
                  'laminatHeight':2., 'numberOfLayers':int(8),
                  'numberMeshWidth':int(21), 'numberMeshLength':int(7), 'numberMeshHeight':int(1),
                  'Lage_1Z':0.0, 'Lage_2Z':0.0,  'Lage_3Z':0.0, 'Lage_4Z':0.0,
                  'Lage_5Z':0.0, 'Lage_6Z':0.0,  'Lage_7Z':0.0, 'Lage_8Z':0.0,
                  'Lage_1Y':0.0, 'Lage_2Y':0.0,  'Lage_3Y':0.0, 'Lage_4Y':0.0,
                  'Lage_5Y':0.0, 'Lage_6Y':0.0,  'Lage_7Y':0.0, 'Lage_8Y':0.0,
                  'Lage_1X':0.0, 'Lage_2X':0.0, 'Lage_3X':0.0, 'Lage_4X':0.0,
                  'Lage_5X':0.0, 'Lage_6X':0.0, 'Lage_7X':0.0, 'Lage_8X':0.0,
                  'E_L':163000.0, 'E_N':8500.0,
                  'G_LN':4200.0, 'G_LT':4200.0, 'G_TN':3360.0,
                  'NU_LN':0.35, 'NU_LT':0.35, 'NU_TN':0.26,
                  'DISPL':-0.1 }
```

Abbildung 15: Auszug aus Python Programm: Dictionary

In Abbildung 16 ist die Definition der Parameter nach der Morris Methode für die Parameter des Lagenwinkels der Lage 8, die Lagendicke, die Probengeometrie und die Verschiebung zu sehen.

```
parmNormalDist12 = ot.Uniform(0.,1.)
parmNormalDist12.setName('Normalverteilung von Lage_8Z')
parmNormalDist12.setDescription(['Lage_8Z' ])

#Lagendicke
parmNormalDist13 = ot.Uniform(0.,1.)
parmNormalDist13.setName('Normalverteilung von Laminatdicke')
parmNormalDist13.setDescription(['laminatHeight' ])

#Probengeometrie
parmNormalDist14 = ot.Uniform(0.,1.)
parmNormalDist14.setName('Normalverteilung von Breite')
parmNormalDist14.setDescription(['specimenWidth' ])

parmNormalDist15 = ot.Uniform(0.,1.)
parmNormalDist15.setName('Normalverteilung von Laenge')
parmNormalDist15.setDescription(['specimenLength' ])

#Verschiebung
parmNormalDist16 = ot.Uniform(0.,1.)
parmNormalDist16.setName('Normalverteilung von Verschiebung')
parmNormalDist16.setDescription(['DISPL' ])
```

Abbildung 16: Auszug aus dem Python-Programm: Definition der Paramater mit deren Verteilungsfunktionen

Um die Rechenzeit zu verkürzen wird die Parameter-Anzahl verringert. Dafür werden die einzelnen UD-Schichtdicken zu einer Laminatdicke verschmiert und es ergeben sich 16 Eingangsparameter. Die vier Material-Parameter (E_1, E_2, G_{12}, ν_{12}), acht Lagenorientierungsparameter (Lage_1 bis Lage_8), zwei Probengeometrieparameter (Breite und Länge), die Laminathöhe und der Verschiebungsparameter werden eingegeben. Nach der Definition und Zuweisung der Variablen werden die einzelnen Verteilungsfunktionen zu einer einzigen zusammengefasst. Dies geschieht mit der Copula-Theorie.

Ist die gemeinsame Verteilungsfunktion berechnet, werden die Versuchspläne erstellt. Die 16 unabhängigen Parameter bedeuten eine sehr hohe Rechenzeit für den Versuchsplan mit den Sobol-Indizes. Um deswegen einige Parameter kürzen zu können, wird erst der Versuchsplan für die Morris-Theorie erstellt und die

Sensitivitäten der einzelnen Unsicherheiten werden berechnet. Diese benötigt eine geringere Rechenzeit und es können die Parameter herausgefiltert werden, welche einen nur geringen Einfluss auf das Ergebnis haben. Die Erstellung der einzelnen Verteilungsfunktionen, das Zusammenfassen der Verteilungsfunktion und das Erstellen der Versuchspläne werden mit der Applikation von OpenTURNS realisiert. Wie schon erwähnt, ermöglicht OpenTURNS mathematische Rechnungen in das Programm einzubinden und diese zu berechnen. Wie OpenTURNS die Rechnungsoperationen durchführt und welche weiteren Funktionen OpenTURNS bietet, kann in der Literatur (Michaël Baudin, 2015) nachgeschlagen werden.

Im folgenden Schritt wird die Datei "GeometrieVernetzung.py" erstellt. Dies geschieht mit dem Python-Programm. In der Datei sind Angaben über die Geometrie, die Lagenanzahl und die Vernetzungsart enthalten. Um die Geometrie zu vernetzen, wird die Vernetzungsdatei "Case_1.med" mit Salome Meca gestartet und die zuvor erstellte "GeometrieVernetzung.py"-Datei wird eingelesen. Daraufhin erstellt Salome Meca mit den Informationen aus der "GeometrieVernetzung.py"-Datei die erforderliche Vernetzung und die zugehörige Vernetzungsdatei "Case_1.med" und speichert sie ab. Anschließend wird Salome Meca geschlossen.

Für das Berechnungsprogramm Code_Aster sind drei Dateien nötig: "Case_1.med", "Case_1.export" und "Case_1.comm". "Case_1.med" wird von Salome Meca erstellt, während die "Case_1.export"- und "Case_1.comm"-Dateien mit dem Python-Programm erstellt werden. Die "Case_1.export"-Datei enthält Informationen über den zur Verfügung gestellten Speicherplatz und die Speicherorte der einzelnen Dateien, die ausgegeben werden sollen. Die "Case_1.comm"-Datei enthält Informationen über die Berechnungsart und definiert die Finiten-Elemente. Ebenso weist es die Lasten und Randbedingungen zu. Sind diese erstellt und stehen zur Verfügung, kann Code_Aster gestartet werden. Die Dateien werden eingelesen und im Anschluss werden die Rechnungen durchgeführt. Nach Abschluss der Rechnungen werden die Spannungen in den einzelnen Schichten ausgegeben. Im Anschluss werden die Knotenkräfte und Spannungen in den einzelnen Schichten Tabellarisch aufgezeichnet ausgegeben.

Zusätzlich zu dem Programm werden die Festigkeiten untersucht. Um diese mit deren Unsicherheiten einzubeziehen, wird das Versagenskriterium "First-Ply-Failure" angewandt. Dieses Kriterium zeigt an, wenn die Maximalspannung in einer der Schichten erreicht ist, und wird für jede einzelne modellierte Schicht angewandt.

Für dieses Kriterium werden die dafür benötigten Daten aus der Sensitivitätsanalyse, welche mit den Sobol-Indizes erfolgt, verwendet. Es werden die Materialauslastungen mit der minimalen Festigkeitsabweichung und maximalen Festigkeitsabweichung ($\pm$5%) berechnet. Diese Rechenoperation wird wieder mit OpenTURNS durchgeführt. Das Resultat dieser Berechnung ergibt die jeweilige Materialauslastung.

3.2 Erstellen der Geometrie und Vernetzung mit Salome Meca

Für das Erstellen der Geometrie wird der Präprozessor Salome Meca verwendet. Ein Präprozessor verarbeitet die Eingabedaten und gibt diese weiter an den Rechenprozessor. In diesem Fall gibt er die Vernetzungsdatei an Code_Aster weiter. Salome Meca ist aufgebaut wie viele andere 3D-Zeichenprogramme, hat aber den Vorteil, dass eine eigene Vernetzungssoftware integriert ist. Außerdem ist Code_Aster, der Rechenprozessor, integriert. Für den Postprozessor bietet Salome Meca die Applikation ParaView an. Ein Postprozessor ist ein Programm, das die Ergebnisse des Rechenprozessors visualisiert. Es werden die Spannungen, beispielhaft in der Abbildung 17 dargestellt, Deformationen und Knotenkräfte farblich an dem Bauteil veranschaulicht.

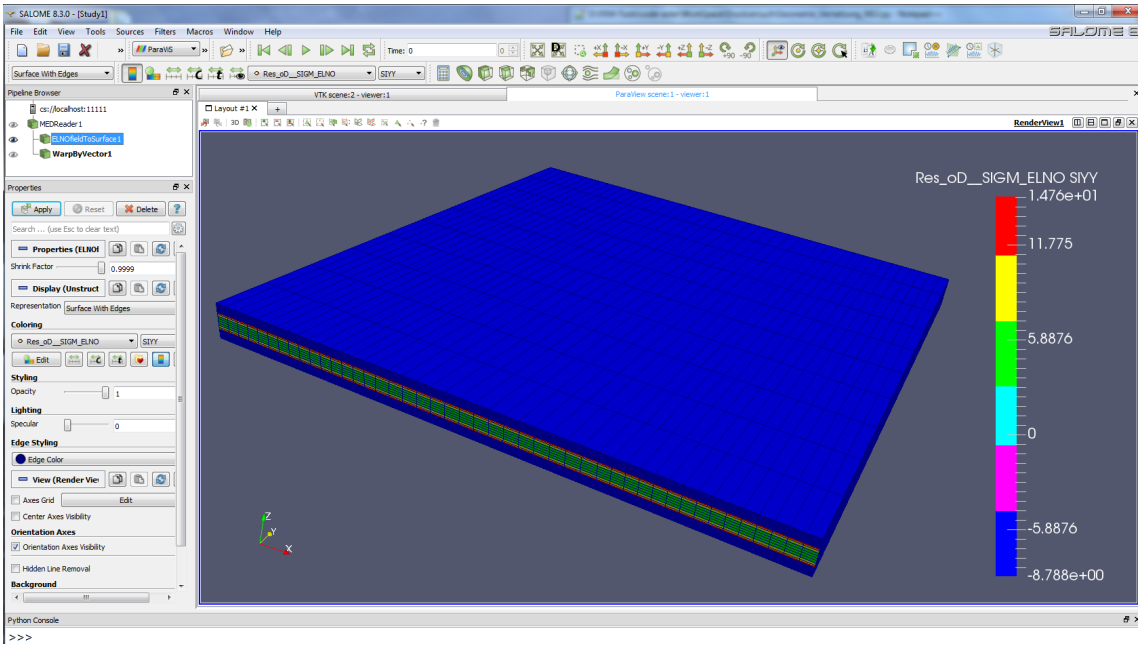


Abbildung 17: ParaView Spannungsdarstellung

Bei der Modellbildung der Geometrie können die Befestigungsflächen vernachlässigt werden, da sie die Kräfte im Idealfall ohne Verlust auf das freie Volumen übertragen. Es wird, wie in Abbildung 18 zu sehen ist, das rot markierte Volumen modelliert. Durch diese Vereinfachung sind weniger Knoten zu berechnen und somit wird die Berechnungszeit verringert. Es wird ein Laminat mit dem Außenmaß 22 mm x 22 mm, einer Lagendicke von 0,25 mm und acht Lagen erzeugt. In Abbildung 19 ist die erstellte Geometrie mit den Partitionen zu sehen. Die Partitionen werden für die späteren Berechnungen der Spannungen benötigt. Um diesen Vorgang später automatisieren zu können, werden die Maße mit Variablen versehen und in das Python Dictionary aufgenommen.

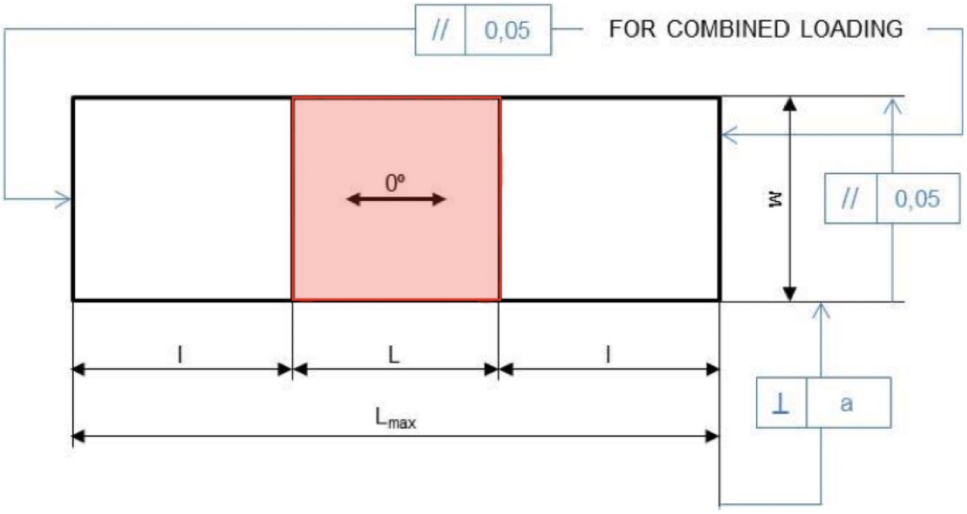


Abbildung 18: Modellierung der Probe

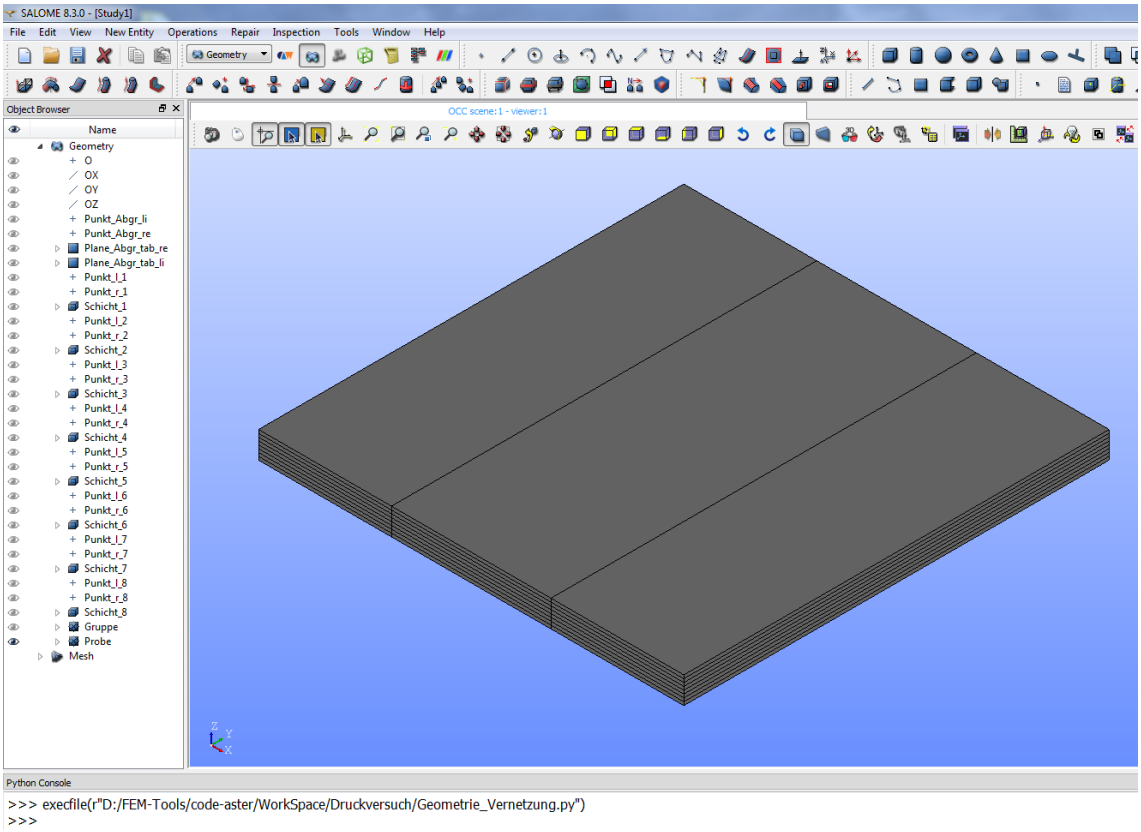


Abbildung 19: Mit Salome Meca erstellte Geometrie

Nach der Fertigstellung der Geometrie müssen noch Volumen, Flächen, Kanten und Knoten in Gruppen zusammengefasst werden. Diese werden für die spätere Vernetzung und das Erstellen der Randbedingungen und Lasteinleitung benötigt. Für die Vernetzung werden drei Kanten entlang der drei Achsen, in einer Gruppe zusammengefasst. Dies ist notwendig, um eine automatisierte und variable Vernetzung zu gestalten. Um die Geometrie und die Lagenanzahl variabel zu

modellieren, müssen diese Vorgänge automatisiert ablaufen. Dabei hilft es, dass das Programm bei der Zählung der einzelnen Volumenkörper, Flächen, Ecken und Knoten im Ursprung beginnt, denn somit kann durch einen einfachen Zähl-Algorithmus auf die einzelnen Elemente zugegriffen werden. Das Programm wurde so gestaltet, dass die benötigten Elemente und Gruppen trotz der variablen Geometrie und Lagenanzahl richtig zugeordnet und für den späteren Aufruf richtig definiert werden.

Sind alle Elemente definiert, kann in das Vernetzungstool von Salome Meca (Abbildung 20) gewechselt werden. Für die Vernetzung für 3D Köper stehen Hexaeder- und Tetraeder-Elemente zur Verfügung. Tetraeder-Vernetzungen eignen sich hervorragend für eine automatische und schnelle Vernetzung, sind aber für einen reinen Druckversuch ungeeignet, da sie durch ihre Form eine Biegung bei Verschiebungsübertragungen hervorrufen. Außerdem beschreiben sie Verzerrungen nicht so genau wie die Hexaeder-Elemente. Hexaeder-Elemente sind bei komplexen Geometrien und automatischen Vernetzungen jedoch schwer zu vernetzten. Das Vernetzungstool von Salome Meca kann eindimensionale bis dreidimensionale Geometrien vernetzen. Bei komplexen Geometrien ist eine automatische Vernetzung mit Salome Meca nicht möglich und fehlerhaft. Deshalb wurden in der Geometrie-Erstellung die einzelnen Ecken in einer Gruppe zusammengefasst. Dies ermöglicht es, den Prozess zu automatisieren. Um die Geometrie zu vernetzen, wird eine automatische Vernetzung gewählt, welche zwei Subvernetzungen beinhaltet. Auf diese Weise wird eine automatisierte Vernetzung auf allen möglichen Geometrie-Variationen und Anzahl der Lagen ermöglicht.

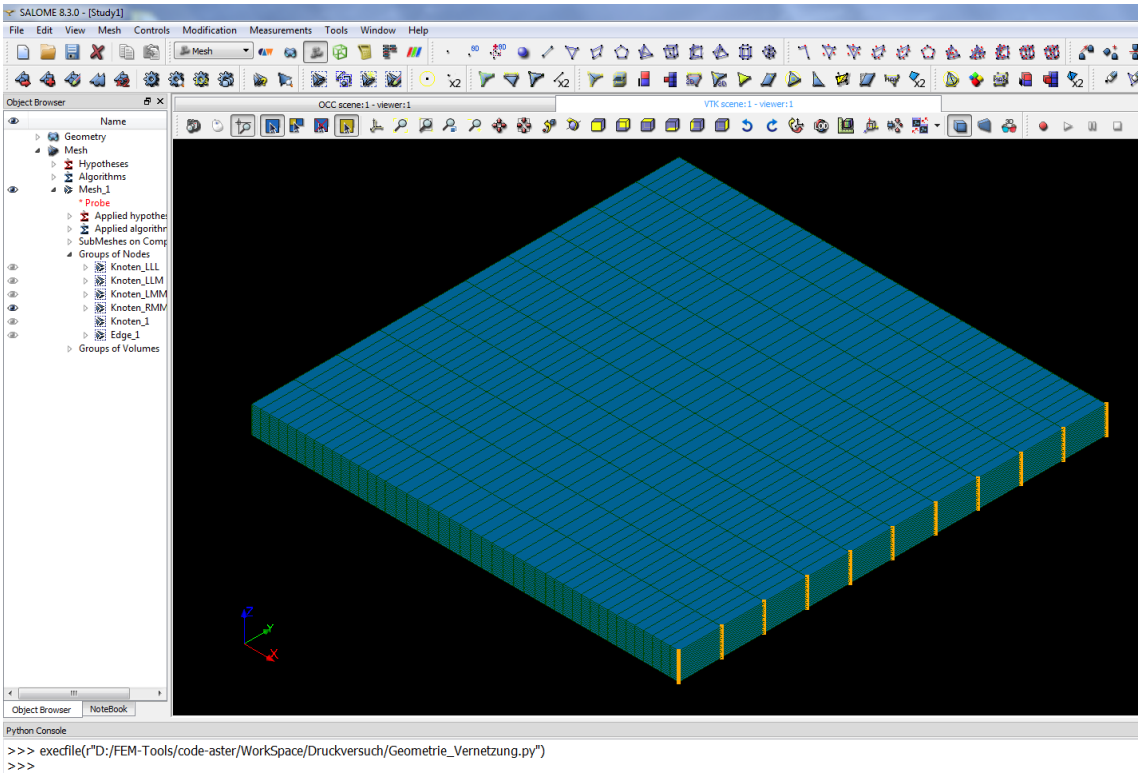


Abbildung 20: Salome Meca vernetzte Geometrie

Für die Konvergenzstudie wird die Anzahl der Finiten-Elemente (FE) in einer Lage in allen drei Koordinaten-Richtungen parametrisiert und automatisiert. Dafür wurden Variablen erstellt, die eine Änderung der Anzahl der FE ermöglichen. Die Variablen wurden in das Python Dictionary aufgenommen. Im Vernetzungstool gibt es Funktionen, um die vorher definierten Volumenkörper, Flächen, Kanten und Knoten zu übertragen. Dies wird angewandt, um die in dem Geometrietool erstellten Gruppen in das Vernetzungstool zu übertragen. Sind alle Gruppen definiert, wird die Vernetzungsdatei ausgegeben.

Als nächstes wird die Konvergenzstudie durchgeführt. Diese wird durchgeführt um das Verhältnis zwischen der Exaktheit des Ergebnisses und der Dauer der Berechnungen zu erhalten und so die Auswahl dieser besser treffen zu können. Mit jeder Erhöhung der Anzahl von FE nähert sich der Ergebniswert der exakten Lösung weiter an und das Ergebnis wird präziser. Dem gegenüber steht jedoch die exponentielle Erhöhung der Berechnungszeit. Wie in Abbildung 21 dargestellt, wurde eine Konvergenzstudie für das hier angewandte Modell durchgeführt. Die Anzahl der Freiheitsgrade (X-Achse) ergeben sich aus der Anzahl der Knoten multipliziert mit den drei Freiheitsgraden, die ein FE besitzt. Zudem ist in dem Diagramm die Berechnungszeit (rechte Y-Achse) abgebildet. Die Berechnungszeit beginnt mit dem Start von Salome Meca und endet mit dem Schließen von Code_Aster. In der Zeit dazwischen erfolgt die Geometrie Erzeugung, die Vernetzung der Geometrie und die Ausgabe der Vernetzungsdatei mit Salome Meca. Zudem das einlesen der Vernetzungsdatei, die Berechnung der FE und die anschließende die Ausgabe der Spannungen in den einzelnen Schichten und Knotenkräfte mit Code_Aster.

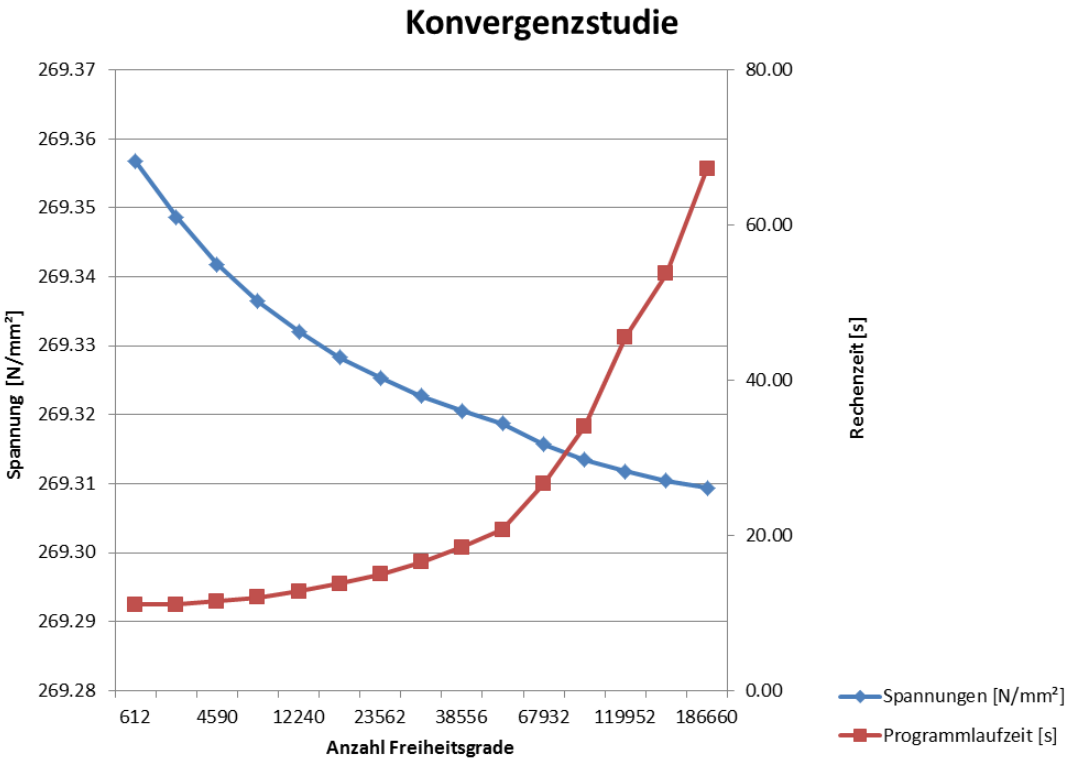


Abbildung 21: Konvergenzstudie

Da sehr viele Berechnungen für die Parameterstudie durchgeführt werden müssen, ist die Wahl auf eine Vernetzung mit 21 Elementen in der Länge, 21 Elementen in der Breite und einem Element pro Lage gefallen. Hierfür wird eine Berechnungszeit von ca. 15 Sekunden pro Rechnung gewählt. Dabei beansprucht das Öffnen und Schließen von Salome Meca, mit ca. 10 Sekunden die meiste Zeit.

3.3 Erstellen von Berechnungen mit Code_Aster

Für die Berechnung der Druckprobe wird Code_Aster verwendet, ein kostenfreies Open-Source-Programm. Die Programmiersprache von Code_Aster basiert auf Python-Befehlen, lediglich eigene Kommando-Befehle werden bei Code_Aster verwendet. Dieses Programm besitzt keine eigene grafische Benutzeroberfläche (GUI), sondern diese wurde von Salome Meca entwickelt und übernommen, um die Benutzung des Programms zu vereinfachen. Die GUI von Code_Aster ist in der Abbildung 22 dargestellt.

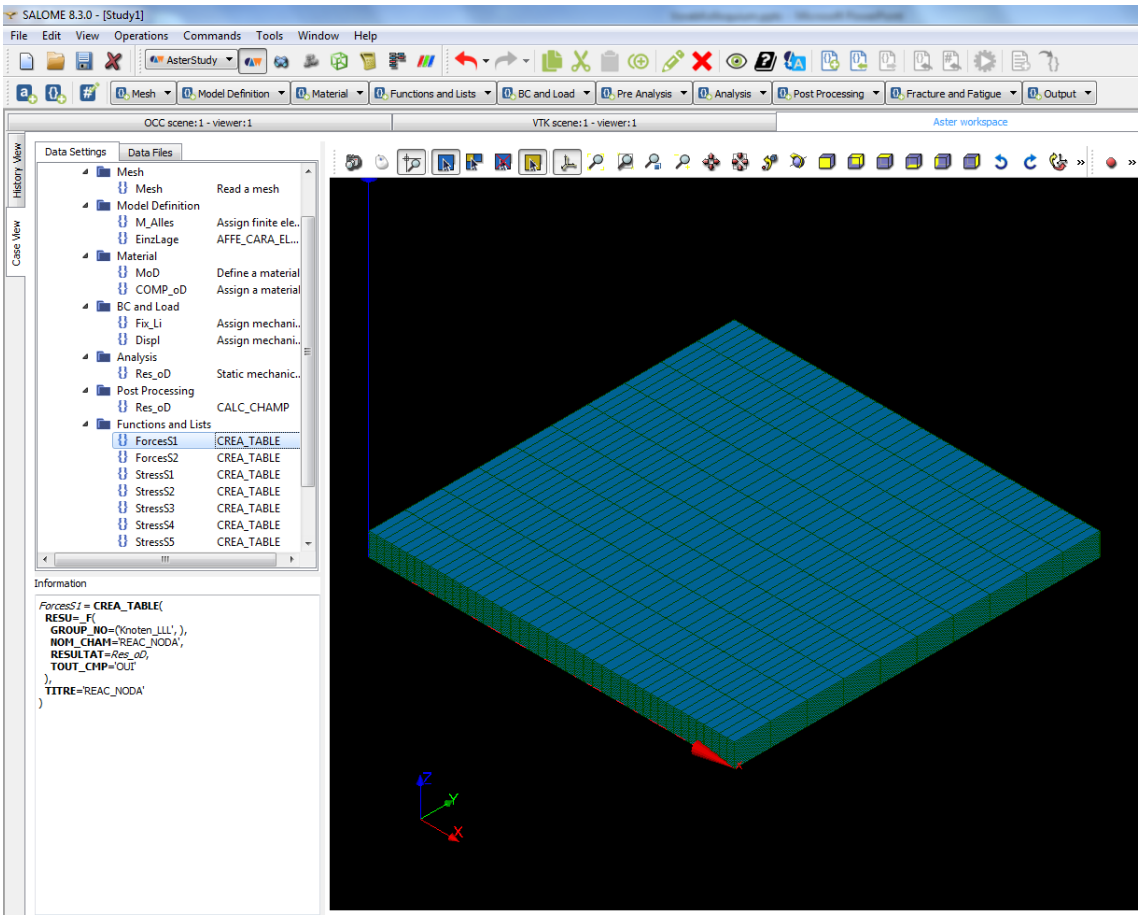


Abbildung 22: Code_Aster Grafische Oberfläche

Es gibt zwei Möglichkeiten die Kommandobefehle über die GUI einzugeben: die Textmethode und die grafische Methode. Mit der Textmethode können die Befehle in Textform eingegeben werden. Um im Anschluss die Angaben auf Richtigkeit zu überprüfen, kann in die grafische Methode gewechselt werden. In der grafischen Methode, welche in Abbildung 22 zu sehen ist, werden die einzelnen Befehle aufgerufen und in einem sich neu öffnenden Fenster, können einzelne Einstellungsmöglichkeiten ausgewählt und eingegeben werden.

Zuerst wird mit Code_Aster die Vernetzungsdatei eingelesen. Hier besteht die Möglichkeit sie aus einem Pfad einzulesen oder die zuvor in Salome Meca erstellte Vernetzung zu übernehmen. Anschließend werden die FE definiert. Code_Aster ermöglicht die Zuweisung von ca. 160 mechanischen, zehn thermischen und zwei akustischen Eigenschaften.

Für die Modellierung eines FV gibt es in Code_Aster bereits eine wählbare Applikation, welche aber nur auf Schalen-Elemente anwendbar ist. In dieser Arbeit werden jedoch 3D-Elemente verwendet und die Winkelorientierungen der Fasern werden durch das Drehen der jeweiligen lokalen Koordinatensysteme ermöglicht. Somit ist die Grundlage geschaffen, für jede einzelne UD-Schicht den Faserablagewinkel zu definieren. Es wird ein Faserverbund mit acht Lagen modelliert, die alle einen Winkel von 90° aufweisen.

Im nächsten Schritt werden die Materialeigenschaften zugewiesen. Code_Aster bietet eine große Auswahl an bereits zur Verfügung stehenden Materialeigenschaften an. Um ein transversales isotropes Materialverhalten zu modellieren, wird das orthotrope Materiaverhalten bei Code_Aster gewählt. Bei dieser Auswahl sind Werte für folgende Variablen einzutragen: E_L, E_N, E_T, G_LN, G_LT, G_TN, NU_LN, NU_LT, NU_TN. Dabei sind E_L = E_1, E_N, E_T = E_2, G_LN und G_LT = G_{12}, NU_LN und NU_LT = ν_{12}, und NU_TN = ν_{23}. Für diese Werte werden die Größen aus dem zur Verfügung gestellten Datenblatt verwendet und eingetragen.

Code_Aster bietet die Möglichkeit, einem Bauteil mehrere Materialien zuzuordnen. So könnte jeder einzelnen Schicht ein anderes Material zugewiesen werden. Diese Funktion bietet sich für eine Modellierung eines Defektes in einer einzelnen Schicht gut an. Dafür wird eine Schicht mit verminderten Eigenschaften modelliert, welche anschließend der defekten Schicht zugewiesen wird.

Im nächsten Schritt werden die einzelnen Randbedingungen und Lasten definiert. Da in der Vernetzung mit Salome Meca vorher die einzelnen Gruppen bestimmt worden sind, können sie jetzt einfach aufgerufen werden um deren Freiheitsgrade zu definieren. Alle sechs Freiheitsgrade, die ein Knoten besitzt, müssen definiert werden, da es sonst zu Konflikten in den Berechnungen kommt. Es sind drei Verschiebungsfreiheitsgrade und drei Verdrehungsfreiheitsgrade zu definieren. Um die Verdrehung um die Y- und Z-Achse und die Verschiebung in X-Richtung zu verhindern, werden die Knoten an der linken Stirnseite (gelb markiert in Abbildung 23) gegen das Verschieben in X-Richtung gesperrt. Und somit ist automatisch auch eine Verdrehung um die Y-Achse und um die Z-Achse verhindert.

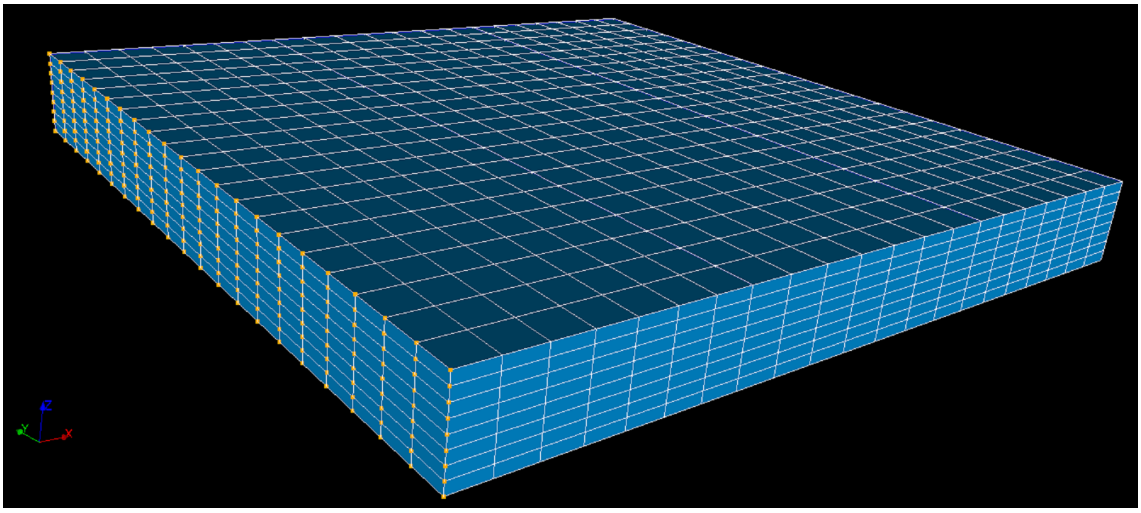


Abbildung 23: Code_Aster Knoten an der Stirnseite

Gegen die Verschiebung in Z-Richtung und eine Verdrehung um die X-Achse wird für die untere Kante (gelb markiert in Abbildung 24) der Stirnseite die Verschiebung in Z-Richtung gesperrt und somit ist auch automatisch die Verdrehung um die X-Achse blockiert.

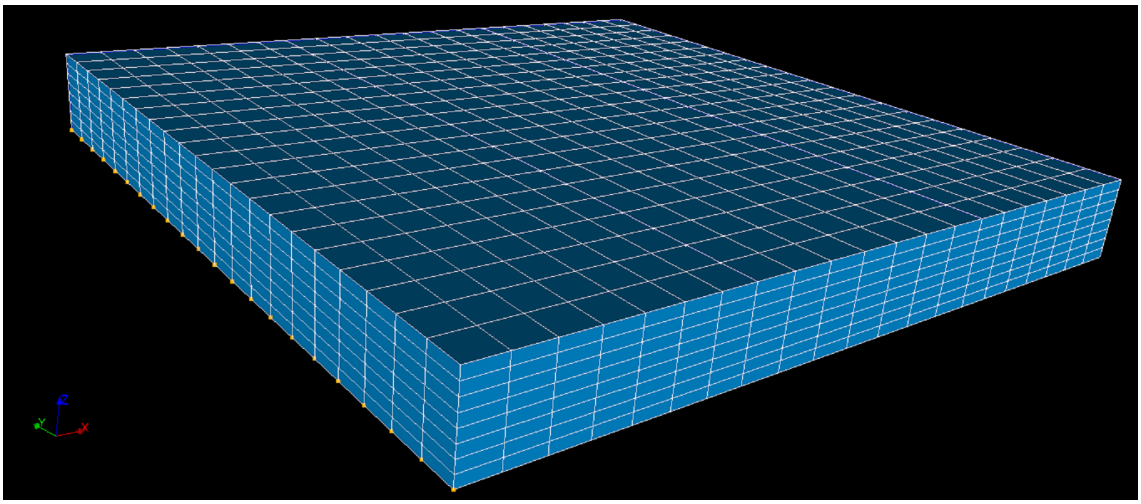


Abbildung 24: Code_Aster Untere Kante

Für den letzten Freiheitsgrad, die Verschiebung in Y-Richtung, wird ein Knoten, welcher im Ursprung liegt (gelb markiert in Abbildung 25), gegen die Verschiebung in Y-Richtung gesperrt.

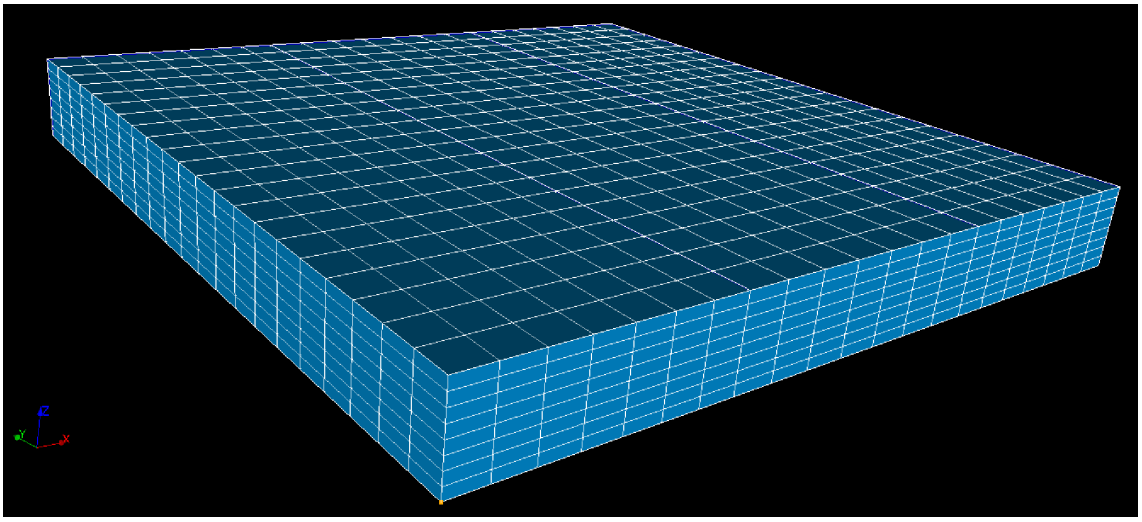


Abbildung 25: Code_Aster Punkt

Für die nun folgende Lasteinleitung werden alle Knoten auf der rechten Stirnseite ausgewählt und es wird eine Verschiebung in negative X-Richtung eingeleitet. Nach diesem Schritt sind alle Randbedingungen und Lasten definiert.

Es erfolgt jetzt die Auswahl, welche Analysenart durchgeführt werden soll. Zur Verfügung stehen lineare und nichtlineare Analysen, dynamische und statische Analysen sowie Vibrationen und thermische Modellierungen. Da in Code_Aster für eine nichtlineare Analyse elastische orthotrope Materialen keine Versagenskriterien zur Verfügung stehen, kann bei dem Umfang dieser Arbeit nur eine statisch lineare Analyse erfolgen. Code_Aster bietet jedoch die Möglichkeit, einen zusätzlichen Programmcode mit einzubinden. So können nach Bedarf und Anforderung eigene Versagenskriterien entwickelt und in Code_Aster eingebunden werden.

Um die, durch die Randbedingungen entstehenden, Fehler nicht in die Ergebnisse mit einfließen zu lassen, werden nur die Spannungen der mittleren Elemente (weiß markiert in Abbildung 26) ausgewertet.

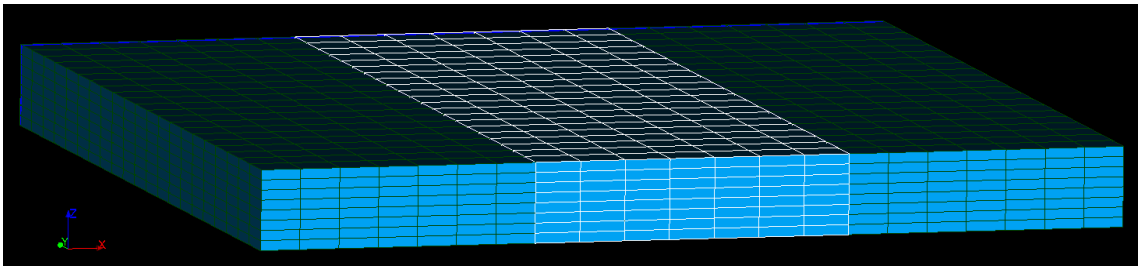


Abbildung 26: Code_Aster Volumen für Spannungsauswertung

Für die Auswertung der auftretenden Kräfte werden die Kräfte, die an den Knoten von der Stirnseite wirken, ausgewertet und ausgegeben. Für die Ausgabe dieser Kräfte werden die Kräfte an den beiden Stirnseiten berechnet und in einzelnen Tabellen ausgegeben. Die Kräfte an den Stirnseiten müssen identisch sein. Für

die Ausgabe der Spannungen in den einzelnen Schichten wird jeweils eine Tabelle erstellt, welche die berechneten Spannungen für die einzelnen Schichten herausfiltert und in einer gesonderten Datei abspeichert. Dieser Prozess wurde ebenfalls parametrisiert und automatisiert. Somit ist bei Eingabe von mehreren Lagen gewährleistet, dass das Programm alle Schichten einzeln ausgibt. Die Spannungen werden in den Koordinaten des globalen Koordinatensystems ausgegeben und müssen, wenn der Lagenwinkel nicht 0° beträgt, noch in das jeweilige lokale Koordinatensystem transformiert werden. Dies geschieht über eine Applikation von OpenTURNS und wird in das Programm eingebunden. Die beiden Ergebnisse für die Spannungen, global und lokal, werden für jedes Element einzeln ausgegeben.

Code_Aster hat die Möglichkeit alle Kommandobefehle, die in der GUI eingegeben worden sind, auszugeben und in eine gesonderte Datei zu schreiben und zu speichern. Daraus wird die Datei „Case_1.comm“ abgeleitet.

Für die Automatisierung des Prozesses benötigt Code_Aster, wie bereits erwähnt, drei Dateien: „Case_1.comm“, „Case_1.med“ und „Case_1.export“. Die „Case_1.export“-Datei wird in der GUI automatisch erstellt. Über diese erstellte Datei kann der erforderliche Inhalt der "Case_1.export"-Datei entnommen werden und mit dem Python-Programm erstellt. Die Datei "Case_1.export" enthält unter anderem folgenden Angaben:

- Wie viel Speicherplatz für die Rechnung zur Verfügung stehen wird
- Wie viele CPUs verwendet werden können
- Temporärer Speicherort
- Wie viel Zeit für eine Rechnung zugeteilt wird
- Speicherorte der Ausgabedateien

4 Parameterstudie am Beispiel einer Druckprobe

4.1 Sensitivitätsanalyse mit der Morris-Methode

In diesem Abschnitt der Arbeit geht es um die Sensitivitätsanalyse nach der Morris-Methode. Hierbei werden 16 Unsicherheiten bzw. Parameter eingesetzt und anschließend wird deren Sensitivität auf das Ergebnis berechnet und bezogen. Dabei werden für einen Parameter (k) 15 Berechnungen (r) durchgeführt. Somit ergeben sich nach der Gleichung (19):

$$\mathrm{n} = \mathrm{r} * \mathrm{k} = 15 * 16 = \mathbf{240}$$

240 Rechnungen (n). Für jede Rechnung (n) wird eine Berechnungszeit von 15 Sekunden benötigt. Wird nun die Zeit mit der Anzahl der Rechnungen (n) multipliziert, dann ergibt sich eine Zeit von 3600 Sekunden also 60 Minuten.

Nach der Berechnung werden für jede Unsicherheit zwei Werte ausgegeben: der Mittelwert (μ) und die Standardabweichung (σ). Diese Ergebnisse wurden in einem Punktdiagramm (Abbildung 27) zusammengefasst.

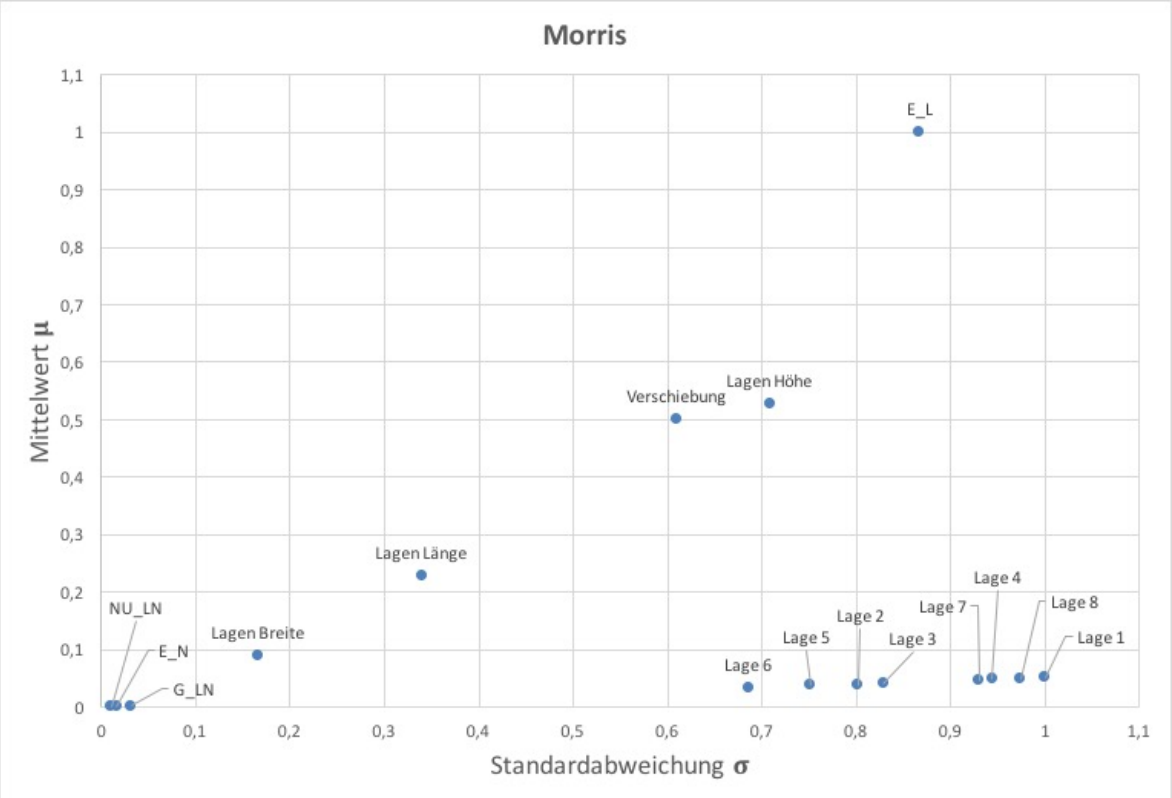


Abbildung 27: Sensitivitätsanalyse Morris-Methode

In diesem Punktdiagramm ist zu erkennen, dass das E-Modul in Faserrichtung E_L (E_1) den größten Einfluss auf die Spannungen besitzt. Der hohe Mittelwert (μ) ist auch durch das hookesche Gesetz

$$\sigma_{L1} = \varepsilon_{L1} * E_1 = \frac{F}{A}$$

abzuleiten, da der E-Modul (E_1) im Produkt mit ε_{L1} die Spannung in Faserrichtung ergibt. Die Winkeländerungen werden durch eine Koordinatentransformation realisiert, was die E-Module dadurch verändert. Demzufolge ist die Wechselwirkung der E-Module mit den Lagenwinkeländerungen sehr hoch und somit auch die Standardabweichung (σ).

Um den Wert des Mittelwertes (μ) bei der Lagenhöhe, -breite und -länge auf die Spannungen erklären zu können, wird das hookesche Gesetz wie folgt umgestellt:

$$F = A * E_1 * \varepsilon = h * b * E_1 * \frac{\Delta l}{l} \tag{21}$$

Hier wird ersichtlich, dass alle drei Faktoren enthalten sind. Wie aus Tabelle 1 zu entnehmen ist, haben Höhe, Breite und Länge unterschiedlich große Toleranzen, die gleichzeitig die jeweilige Einflussgröße auf die Spannung darstellen. Die Lagenhöhe (h) hat mit einem Variationskoeffizienten von 14 % die größte Auswirkung auf den Mittelwert (μ), weswegen diese auch größer ausfällt als die der Lagenlänge (l) und der Lagenbreite (b). Deren jeweilige Variationskoeffizienten betragen nur 2,3 % (l) und 0,9 % (b). Auch der hohe Wert der Standardabweichung (σ) ist nachvollziehbar, da die Parameter eine Wechselwirkung untereinander besitzen.

Über die umgestellte Formel lässt sich ebenso der Wert der Verschiebung (Δl) und dessen hohen Mittelwert (μ) erklären. Durch den Variationskoeffizienten von 5 % nimmt diese den drittgrößten Einflussfaktor ein. Auch hier lässt sich der große Wert der Standardabweichung (σ) mit der Wechselwirkung zu den anderen Parametern der Gleichung (21) erklären.

Der kleine Wert der Mittelwerte (μ) bei den Lagenwinkeln, von Lage 1 bis Lage 8, ist auf die kleinen Toleranzen der Winkelabweichungen zurückzuführen. Somit werden nur kleine Änderungen der E-Module verursacht, welche wiederum kleine Änderungen der Spannungen verursachen. Dass die Standardabweichung (σ) sehr hoch ist, kann damit begründet werden, dass bei Änderungen der Winkel die E-Module direkt verändert werden.

Die kleinen Werte der Querkontraktionszahl NU_LN (ν_{12}), des E-Moduls quer zur Faserrichtung E_N (E_2) und des Schubmoduls G_LN (G_{12}) bei dem Mittelwert (μ) haben kaum Einfluss in Faserrichtung, da diese im hookeschen Gesetz keine Einflussgrößen besitzen und die Einflussgrößen auch nicht indirekt beeinflussen können. Die minimalen Werte dieser Größen lassen sich über die

Winkeländerungen der einzelnen Schichten erklären, da das E-Modul quer zur Faserrichtung (E_2) einen minimalen Einfluss auf die Spannungen in Faserrichtung hätte. Auch der kleine Wert der Standardabweichung (σ) spiegelt deren kleinen Einfluss auf die Spannung wider.

4.2 Sensitivitätsanalyse mit den Sobol-Indizes

Das Folgende betrifft die Verfeinerung der nach der Morris-Methode erlangten Rechenergebnisse. Diese werden mit dem Verfahren der Sobol-Indizes noch weiter verfeinert und die Wechselwirkung der einzelnen Parameter zueinander wird explizit berücksichtigt. Durch die Berücksichtigung der einzelnen Interaktionen zwischen den Parametern ergibt sich eine deutlich komplexere Rechnung als bei der Morris-Methode und die Rechenzeit vervielfacht sich.

Um Rechenzeit einzusparen, werden einzelne Parameter vernachlässigt. Zudem werden zwei Vereinfachungen vorgenommen. Die einzelnen Lagenwinkel der Lagen 1 bis 8 werden zu nur einem Parameter verschmiert und bei der Probengeometrie wird nur die Lagenhöhe (h) berücksichtigt. Zudem werden der E-Modul in Faserrichtung E_L (E_1), der E-Modul quer zur Faserrichtung E_N (E_2), das Schubmodul G_LN (G_{12}), die Querkontraktionszahl (ν_{12}) und die Verschiebung (Δl) einbezogen. Demzufolge finden in dieser Berechnung nur noch sieben Parameter Anwendung.

Für die Sensitivitätsanalyse mit den Sobol-Indizes wird aufgrund der Rechenzeit das Minimum von 500 Kombinationsvarianten der einzelnen Parameter (N) angesetzt. Somit ergibt sich mit den sieben berücksichtigten Parametern (k) nach der Gleichung (20) eine Anzahl von 8000 Rechnungen (n).

$$n = N * (2k + 2) = 500 * (2 * 7 + 2) = \mathbf{8000}$$

Wird nun die Anzahl der Rechnungen (n) mit einer jeweiligen Berechnungszeit von 15 Sekunden multipliziert, ergibt sich eine Gesamtberechnungszeit von 120.000 Sekunden, also 33,33 Stunden. Die Berechnungszeit von 15 Sekunden pro Berechnung (n) wurde durch die im Voraus erfolgte Konvergenzstudie festgelegt. Nach der Berechnung werden der Mittelwert und die Standardabweichung für die First Order (FO) und Total Order (TO) ausgegeben. Ergebnisse für die Second Order (SO) konnten nicht ausgegeben werden, da dafür mehr Kombinationsvarianten der einzelnen Parameter benötigt würden. Die Ergebnisse wurden in dem Diagramm (Abbildung 28) abgebildet.

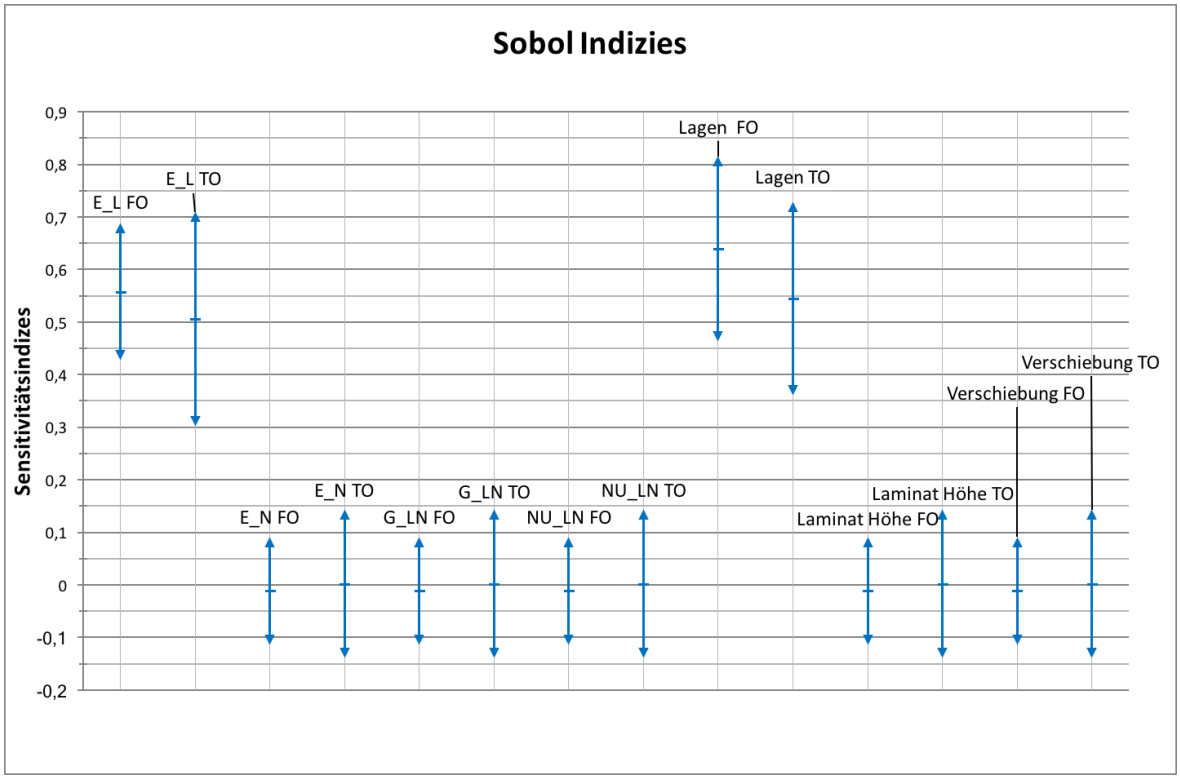


Abbildung 28: Sensitivitätsanalyse mit Sobol-Indizes

Wie in diesem Diagramm zu erkennen ist, könnten die Parameter E_N (E_2), G_LN (G_{12}), NU_LN (ν_{12}), Laminathöhe (h) und die Verschiebung (Δl) fehlerhaft sein, da sie in den negativen Bereich reichen. Zudem sind sehr große Standardabweichungen bei allen Parametern ersichtlich, welche ebenso den Mangel an Kombinationsvarianten zeigen. Wie schon in dem Abschnitt Sensitivitätsanalyse mit der Morris-Methode erklärt, ergibt sich der deutliche Einfluss des E-Moduls in Faserlängsrichtung E_L (E_1) auf den Mittelwert über das hookesche Gesetz. Ebenso verhält es sich mit dem Einfluss der Lagen, die wiederum indirekt über die Lagenwinkeländerung auf den E-Modul einwirken und dieses verringern.

4.3 Auswertungen der Festigkeiten

Um die Materialauslastung bestimmen zu können, werden die Werte verwendet, die für die Unsicherheiten durch den Versuchsplan erstellt worden sind. Mit den berechneten Spannungen und den Festigkeiten, einmal minimal und einmal maximal, werden die Materialauslastungen berechnet. Diese aus dem Versuchsplan vorgegebenen Unsicherheiten werden in reduzierter Form in den sich in diesem Abschnitt befindenden Diagrammen dargestellt, da eine Datenmasse von 8000 Datenpunkten zu unübersichtlich ist. So wurde nur jeder 80. Datenpunkt für die Auswertung genutzt.

Die Festigkeitstoleranzen werden in Abhängigkeit der variierenden Unsicherheiten untersucht. Hierbei werden die Unsicherheiten über die Materialauslastungen dargestellt. Es ergibt sich für die minimale Festigkeitsabweichung das in Abbildung 29 abgebildete Diagramm. Die maximale Festigkeitsabweichung wird in Abbildung 30 dargestellt.

Zur besseren Veranschaulichung wurden die Verschiebung (Δl), der E-Modul quer zur Faserrichtung (E_2) und der Schubmodul (G_{12}) noch einmal in der Abbildung 31 für die maximale Materialauslastung separat betrachtet und skaliert, um die bei den Werten ebenso vorhandenen Schwankungen zu verdeutlichen. Auf die vergrößerte Ansicht dieser Werte für die minimale Materialauslastung wurde verzichtet, da die Darstellungen, wie in den Abbildung 29 und Abbildung 30 zu sehen, analog zueinander verlaufen.

Die Werte unterscheiden sich von der minimalen zur maximalen Materialauslastung in dem Auslastungsbereich und variieren ansonsten nur mit minimalsten Schwankungen.

Abbildung 29 und Abbildung 30 dienen der Veranschaulichung der unterschiedlichen Unsicherheiten zueinander mit deren Schwankungsbereichen. Die steigende Tendenz des E-Moduls in Faserrichtung (E_1) und die berechnete Steifigkeit mit zunehmender Materialauslastung sind deutlich erkennbar. Eine minimale Tendenz nach oben, ist ebenso bei der Lagenhöhe sichtbar. Die minimal abfallende Tendenz der Lagenwinkeländerung scheint fast vernachlässigbar klein.

Die Materialauslastung, welche jeweils auf der X-Achse aufgetragen ist, beginnt bei einer Auslastung der Probe von 27 % und endet bei 75,5 %. Dieser Wertebereich verdeutlich einmal mehr die großen Sicherheitsfaktoren, welche bei FV verwendet werden, da noch viel Spiel nach oben vorhanden ist.

Die Verschiebung (Δl), wie in Abbildung 31 zu sehen, muss in umgekehrter Weise betrachtet werden, da es sich um eine Druckbeanspruchung handelt und so die unteren Werte der Y-Achse eine größere Drucklast darstellen. Dementsprechend ist hier ebenso eine steigende Trendlinie erkennbar. Die Steigung ist bei dem E-Modul quer zur Faserrichtung (E_2) und beim Schubmodul (G_{12}) nicht so deutlich erkennbar, darf aber dennoch nicht vernachlässigt werden. Sobald sich der Lagenwinkel etwas mehr ändert, ist der Eintrag dieser beiden Größen durch die Koordinatentransformation höher. Die übriggebliebenen Unsicherheiten haben keinen deutlichen Einfluss auf die Festigkeit.

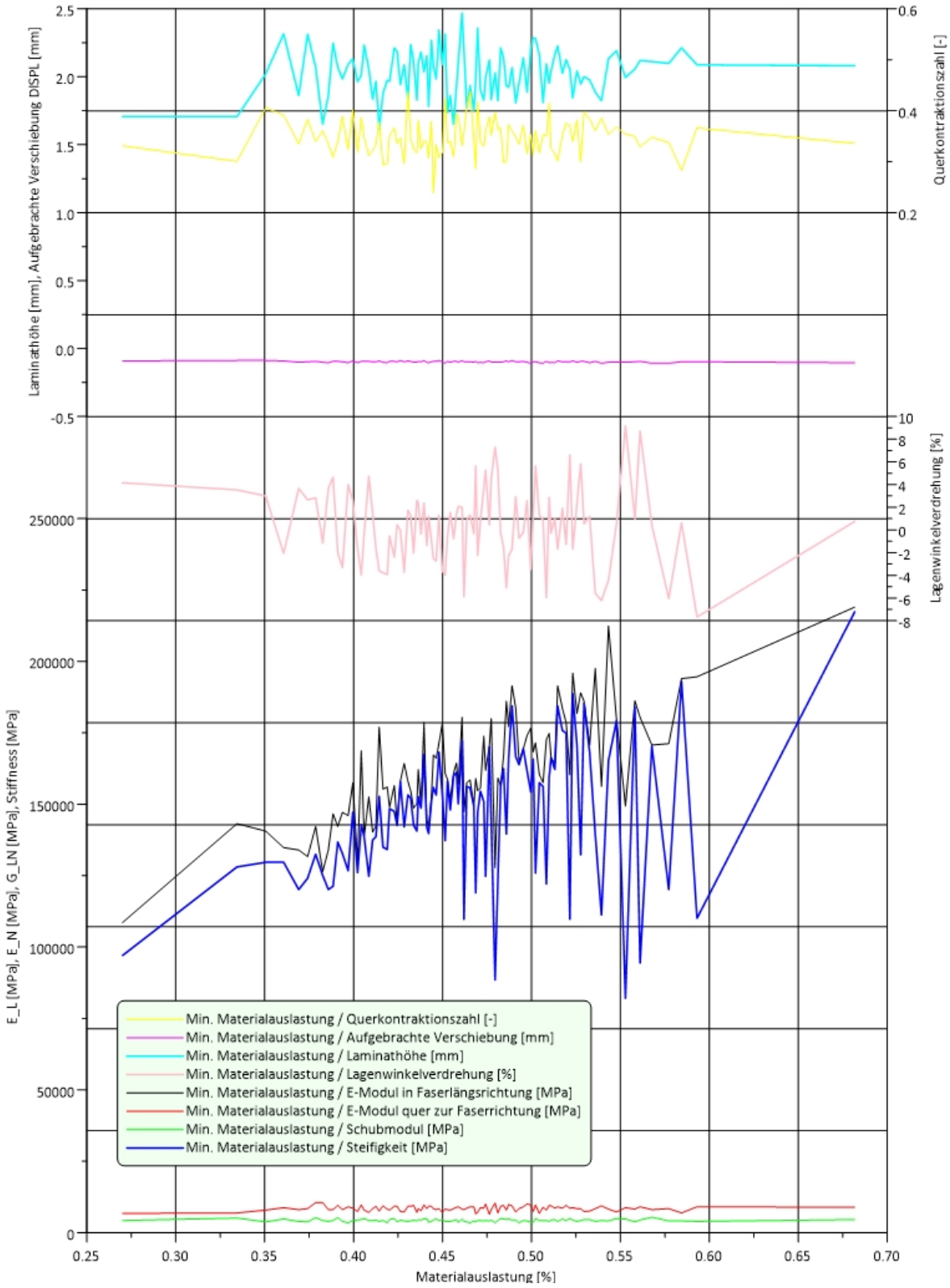


Abbildung 29: Materialauslastung mit minimaler Festigkeitsabweichung

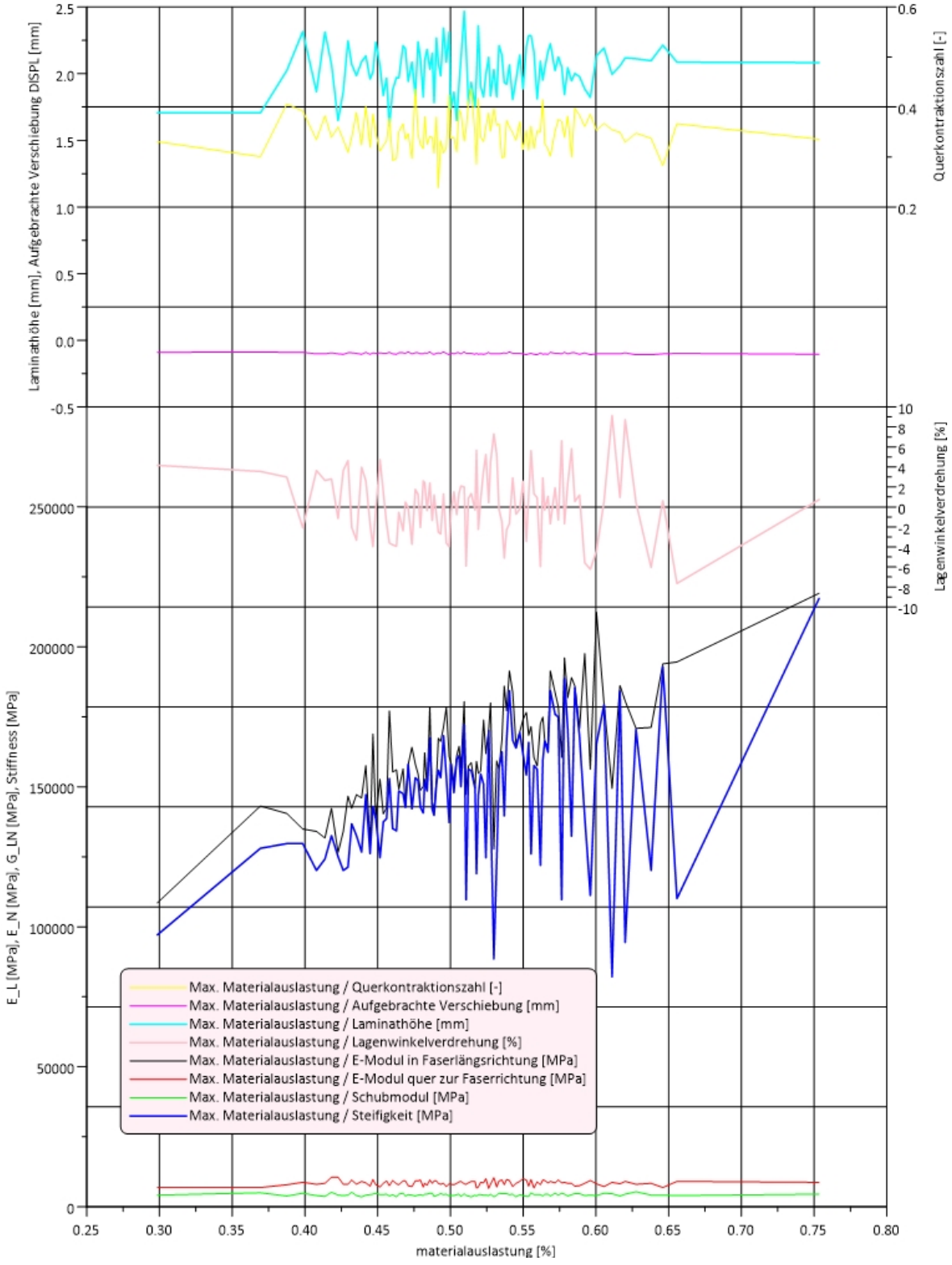


Abbildung 30: Materialauslastung mit maximaler Festigkeitsabweichung

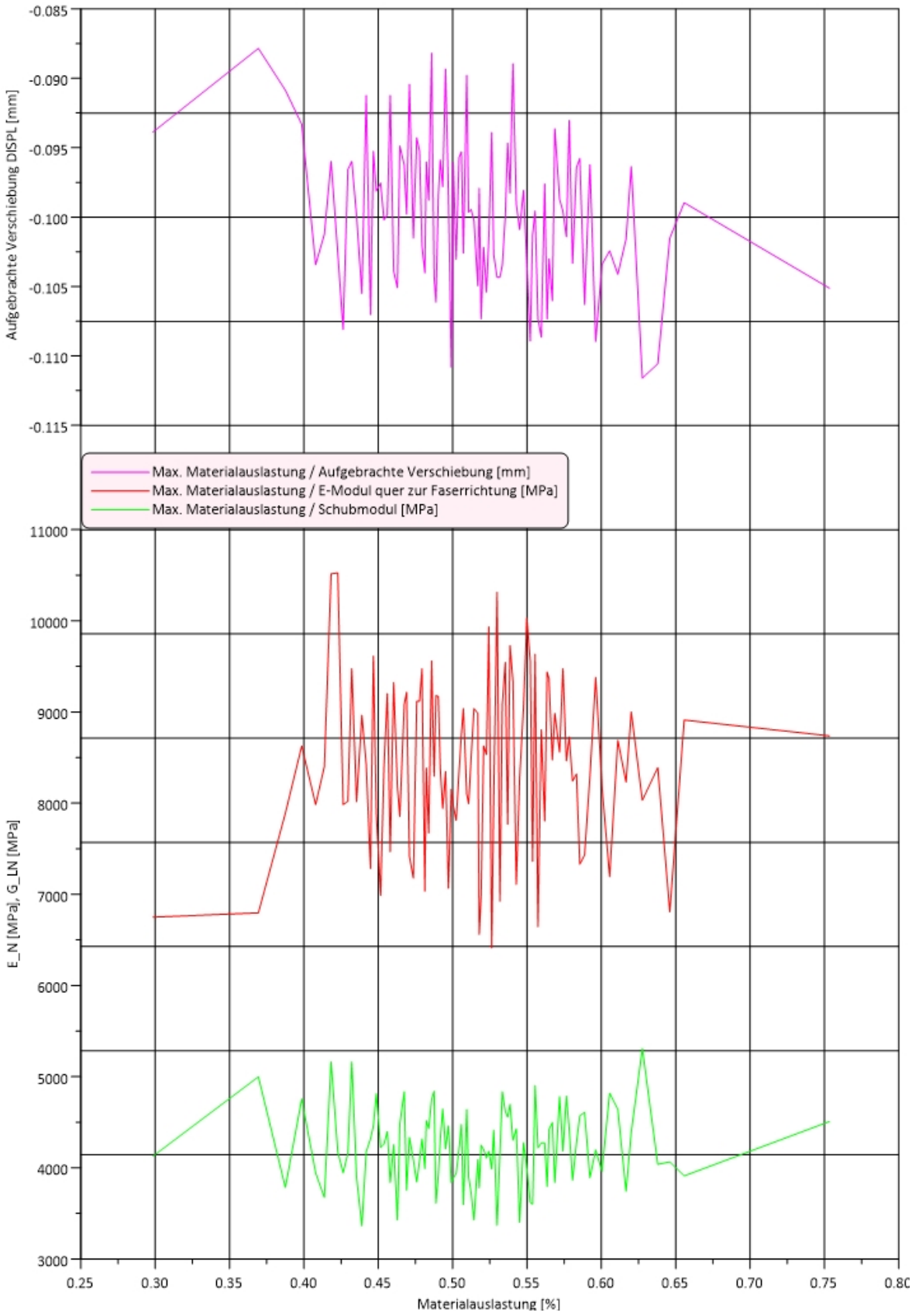


Abbildung 31: Materialauslastung mit maximaler Festigkeitsabweichung

5 Ausblick

Das Ziel dieser Arbeit, die fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen exakter zu bestimmen und somit bewerten zu können, wurde erfolgreich erreicht. Es ist möglich, mit dem Open-Source-Programm Code_Aster lineare und nichtlineare Analysen durchzuführen. Bei den nichtlinearen Analysen ist jedoch zu beachten, dass zusätzliche Subprogramme für die Versagenskriterien geschrieben und eingebunden werden müssen. Die Grundlage, um die Subprogramme einzubinden, bietet Code_Aster. Es stellt sich die Frage, ob sich der Aufwand und die Arbeitszeitkosten im Verhältnis zu den Kosteneinsparungen durch die Verwendung des Open-Source-Programmes Code_Aster im Gegensatz zu den zurzeit verwendeten kostenpflichtigen Programmen rentieren und ob es sich auf demselben Niveau wie diese befindet. Dieser Frage sollte im Anschluss an die aus dieser Arbeit gewonnenen Erkenntnisse weiter nachgegangen werden.

Auch die Steuerung von Salome Meca und Code_Aster über die Kommandozentrale ermöglicht eine einfache Einbindung in einen Berechnungsprozess, ohne sich vorher mit der GUI der beiden Programme auseinandersetzen zu müssen. Deshalb ist eine leichte und schnelle Umstellung auf die Programme möglich.

Anknüpfend an den für diese Arbeit geschriebenen Programmcode kann im weiteren Vorgehen die Erweiterung des Codes um die zusätzlichen wichtigen Geometrieformen des Open hole und des Filled hole folgen.

Weiterführend kann der Programmcode erweitert und ausgebaut werden, um auch thermische Einflüsse und Feuchtigkeitseinflüsse mit in das Programm einzubeziehen und die simulierten Bedingungen noch weiter an die realen Gegebenheiten anzupassen.

Um die Berechnungszeiten weitergehend zu optimieren, wird empfohlen, mit der Morris-Methode die vernachlässigbar kleinen Parameter auszumachen. Dann können diese in der darauffolgenden Berechnung mit den Sobol-Indizes ausgelassen werden. Diese kann theoretisch genauere Ergebnisse liefern, dafür müsste aber die Anzahl der Kombinationsvariation (N) der einzelnen Parameter angehoben werden. Je genauer also die Abstimmung zwischen den beiden Sensitivitätsanalysen ist, desto genauer können die Endergebnisse berechnet werden. Die Rechenzeiteinsparung kann über das Kürzen der Parameteranzahl erfolgen. Die eingesparte Zeit kann anschließend bei der Berechnung der Sobol-Indizes durch eine höhere Anzahl an Rechnungen pro Kombinationsvariante (N) verwendet werden.

Zudem ist zu überprüfen, ob durch genauere Herstellungs- und Fertigungsprozesse, zum Beispiel durch präzisere Maschinen, das Toleranzfeld bei der Probengeometrie verkleinert werden kann.

Die Auswertung der Festigkeiten hat gezeigt, dass die Unsicherheiten einen großen Eintrag in die Sicherheitsfaktoren besitzen. Daher ist zu prüfen, ob eine Verringerung der Schwankungen möglich ist. Durch diese Arbeit hat sich gezeigt,

dass Potenzial in diesem Bereich vorhanden ist, da sich die Materialauslastungen in einem Feld von 27 % bis 75,4 % bewegen.

6 Literaturverzeichnis

AIRBUS AITM, Airbus Test Method, Fibre Reinforced Plastics, Determination of Plain, Open Hole and Filled Hole Compression Strength [Bericht]. - März 2015.

Andrea Saltelli Stefano Tarantola, Francesca Campolongo and Marco Ratto SENSITIVITY ANALYSIS IN PRACTICE [Buch]. - Joint Research Centre of the European Commission, Ispra, Italy : John Wiley & Sons Ltd, 2004.

Antonio Miraglia Danilo Malacaria, Matteo Giugno Virtual Allowables approach for the design phase of a composite fuselage [Bericht]. - [s.l.] : JECMagazine, 08/09.2017.

Caniou Yann Analyse de sensibilité globale pour [Bericht]. - Clermont-Ferrand, France : Université Blaise Pascal - Clermont II, 2012. - 2296.

Caracciolo Paola Multilevel probabilistic approach to evaluate manufacturing defect in composite aircraft structures [Bericht] / Department of Airframe Architecture & Integration ; AIRBUS INDUSTRIES Germany. - Hamburg : AIP Publishing, 2014. - http://dx.doi.org/10.1063/1.4876780.

Company XStream Engineering MSC Software Digimat Virtual Allowables [Bericht]. - 2014.

Danial H.M. Hsiao und I.M. Effect of Fiber Waviness on Stiffness and Strength Reduction of Unidirectional Composites Under CompressiveLoading [Bericht] / Composite Science and Technology 56. - 1996.

Danial H.M. Hsiao und I.M. Elastic Properties of Composites with Fiber Waviness [Bericht] / Composites Part A 27A. - 1996.

Fayazbakhsh Kazem The impact of gaps and overlaps on variable stiffness composites manufactured by Automated Fiber Placement [Bericht] / Department of Mechanical Engineering McGill University, Montreal. - 2013.

Frank Abdi J. Surdenas, Nasir Munir, Jerry Housner and Raju Keshavanarayana Virtual Testing and Predictive Modeling [Bericht]. - [s.l.] : Springer Science+Business Media, 2009.

Holm Altenbach Johannes Altenbach, Roland Rikards Einführung in die Mechanik der Laminat- und Sandwichtragwerke [Bericht]. - [s.l.] : Deutscher Verlaf für Grundstoffindustrie Stuttgart, 1996.

Krimmer Alexander Mikromechanische Modellierung von Fasergelege-Kunststoff-Verbunden auf Basis von Normprüfungen unter Berücksichtigung der in-situ-Eigenschaften der Matrix [Bericht]. - [s.l.] : Universitätsverlag der TU Berlin, 2014.

Mesogitis T.S. Uncertainty in the manufacturing of fibrous thermosetting composites: A review [Bericht]. - [s.l.] : Elsevier, 2013.

Michaël Baudin Anne Dutfoy, Bertrand Iooss and Anne-Laure Popelin OpenTURNS : An industrial software for uncertainty quantification in simulation [Bericht]. - [s.l.] : EDF R&D, 2015.

Nelson Roger B. An Introduction to Copulas [Buch]. - New York : Springer, 1999. - ISBN 978-0-387-98623-4.

Research National Institute for Aviation Hexcel 8552 IM7 Unidirectional Prepreg 190 gsm & 35%RC Qualification Material Property Data Report [Bericht]. - 2011.

Richard Moon KokSiong Lim, Kurt Schueler, Andrew Yoder, Harshinder Singh Integrated Design and Analysis Tools for Reduced Weight, Affordable Fiber Steered Composites [Bericht] / Aerospace Engineering ; University of Kansas. - Kansas : [s.n.], 2003.

Sang Yoon Park Won Jong Choi Production Control Effect on Composite Material Quality and Stability for Aerospace Usage [Bericht]. - 2017.

Sante Raffaella Di MDPI [Online]. - 19. 09 2018. - https://doi.org/10.3390/s150818666.

SOBOL' I. M. Sensitivity Estimates for Nonlinear Mathematical Models [Bericht]. - Russian Academy of Sciences, Moscow : M. V. Keldysh Institute of Applied Mathematics, 1993.

Türck Hans-Friedrich Eckey / Reinhold Kosfeld / Matthias Wahrscheinlichkeitsrechnung und Induktive Statistik [Buch]. - Wiesbaden : Betriebswirtschaftlicher Verlag, 2005. - ISBN 3-8349-0043-5.

Voigt W. Über die Beziehung zwischen den beiden Elastizitätskonstanten isotroper Körper [Bericht]. - Annalen der Physik 274 : [s.n.], 1889.

Wedekind Max Charakterisierung von Steifigkeit und Festigkeit heterogen verstärkter Verbundstrangpressprofile [Bericht] / Maschinenwesen ; Technische Universität München. - 2013.

Xiao Chen Zhiping Qiu A novel uncertainty analysis method for composite structures with mixed uncertainties including random and interval variables [Bericht] / Institute of Solid Mechanics. - [s.l.] : Elsevier, 2017. - http://dx.doi.org/10.1016/j.compstruct.2017.09.068.

Yves Davila Laurent Crouzeix, Bernard Douchin, Francis Collombet, Yves-Henri Grunevald Spatial Evolution of the Thickness Variations over a CFRP Laminated Structure [Bericht] / Université de Toulouse. - [s.l.] : Springer Science+Business Media Dordrecht, 2016. - DOI 10.1007/s10443-016-9573-5.

Anhang A: Python-Code

```
'''
Created on 07.08.2018

@author: hein_fa, loew_pe
'''
from smetana.model.geometry.coordinatesystem import Transformation
import smetana.service.utilities as smetanaUtils
from smetana.failure.stressbased import MCDFibreMaxStress

import openturns as ot
from openturns.viewer import View
import matplotlib.pyplot as plt
# plt.style.use('ggplot')

import otmorris
from otmorris.plot_sensitivity import PlotEE

import subprocess
import os

import numpy as np

def createRunAsterBatch(studyDir, exportFile):
    with open(studyDir+'/asterCall.bat', 'w') as bf:
        bf.write('@ECHO off\n')
        bf.write('\n')
        bf.write('set asterBatchPath=C:\\CodeAsterSolver\\code-aster_v2017_std-win64\\install\n')
        bf.write('set studyDir='+os.path.normpath(studyDir)+'\n')
        bf.write('\n')
        bf.write('cd /d %asterBatchPath%"\\13.4\\share\\aster"\n')
        bf.write('\n')
        bf.write('call %asterBatchPath%"\\bin\\as_run.bat" --run %studyDir%"\\'+exportFile+'"\n')

def createRunSalomeBatch(studyDir, modelFile):
    with open(studyDir+'/salomeCall.bat', 'w') as bf:
        bf.write('@ECHO off\n')
        bf.write('\n')
        bf.write('set salomeBatchPath=D:\\SALOME-Mecha-2017\\WORK\n')
        bf.write('set studyDir='+os.path.normpath(studyDir)+'\n')
        bf.write('\n')
        bf.write('cd /d %salomeBatchPath%\n')
        bf.write('\n')
        bf.write('call %salomeBatchPath%"\\set_env.bat" %salomeBatchPath%"\\run_env.bat"\n')
        bf.write('\n')
        bf.write('call %salomeBatchPath%"\\run_Salome.bat" -t %studyDir%"\\'+modelFile+'"
%salomeBatchPath%"\\kill_Salome.bat"\n')

def callAster(studyDir):
#     ASTER_ROOT='C:/CodeAsterSolver/code-aster_v2017_std-win64/install'
    subprocess.call(studyDir+"/asterCall.bat")

def callSalome(studyDir):
    subprocess.call(studyDir+"/salomeCall.bat")
```

```
    fileExists=False
    while not fileExists:
        if os.path.exists(studyDir+'/Case_1.med'):
            fileExists=True

    return fileExists

def createSalomeModelScript(inputDict):
    print(inputDict)
    with open(studyDir+'/GeometrieVernetzung.py', 'w') as mf:
        mf.write('# -*- coding: utf-8 -*-\n')
        mf.write('\n')
        mf.write('#Das Skript kann zur Ueberpruefung in Salome_Meca eingeladen werden.\n')
        mf.write('#Dafuer einfach Salome_Meca starten, mit "File" "New" neues Projekt starten.\n')
        mf.write('#Anschliesend ueber "File" "Load Skript" diese Datei einlesen.\n')
        mf.write('\n')
        mf.write('import sys\n')
        mf.write('import salome\n')
        mf.write('import salome_notebook\n')
        mf.write('import math\n')
        mf.write('\n')
        mf.write('salome.salome_init()\n')
        mf.write('theStudy = salome.myStudy\n')
        mf.write('notebook = salome_notebook.NoteBook(theStudy)\n')
        mf.write('sys.path.insert( 0, r"'+inputDict['studyDir']+'")\n')
        mf.write('\n\n\n')
        mf.write('###\n')
        mf.write('### specimen geometry\n')
        mf.write('###\n')
        mf.write('\n')
        mf.write('pb='+str(inputDict['specimenWidth'])+' #specimenWidth 32\n')
        mf.write('pl='+str(inputDict['specimenLength'])+' #specimenLength 32\n')
        mf.write('tll='+str(inputDict['tabLengthLeft'])+' #tabLengthLeft 10 \n')
        mf.write('tlr='+str(inputDict['tabLengthRight'])+' #tabLengthRight 10 \n')
        mf.write('al='+str(inputDict['numberOfLayers'])+' #numberOfLayers 8\n')
        mf.write('wlh='+str(inputDict['laminatHeight'])+' #laminatHeight 2\n')
        mf.write('lh=wlh/al #layerHeight 0.25\n')
        mf.write('\n\n\n')
        mf.write('###\n')
        mf.write('### specimen Mesh\n')
        mf.write('###\n')
        mf.write('\n')
        mf.write('anb='+str(inputDict['numberMeshWidth'])+' #numberMeshWidth 10\n')
        mf.write('anh='+str(inputDict['numberMeshHeight'])+' #numberMeshHeight 1\n')
        mf.write('anl='+str(inputDict['numberMeshLength'])+' #numberMeshLength 15\n')
        mf.write('\n\n\n')
        mf.write('###\n')
        mf.write('### GEOM component\n')
        mf.write('###\n')
        mf.write('\n')
        mf.write('import GEOM\n')
        mf.write('from salome.geom import geomBuilder\n')
        mf.write('import SALOMEDS\n')
```

```
mf.write('\n')
mf.write('geompy = geomBuilder.New(theStudy)\n')
mf.write('\n')
mf.write('O = geompy.MakeVertex(0, 0, 0)\n')
mf.write('OX = geompy.MakeVectorDXDYDZ(1, 0, 0)\n')
mf.write('OY = geompy.MakeVectorDXDYDZ(0, 1, 0)\n')
mf.write('OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)\n')
mf.write('\n')
mf.write('tlrr=pl-tlr\n')
mf.write('Punkt_Abgr_li = geompy.MakeVertex(tll, 0, 0)\n')
mf.write('Punkt_Abgr_re = geompy.MakeVertex(tlrr, 0, 0)\n')
mf.write('Plane_Abgr_tab_re = geompy.MakePlane(Punkt_Abgr_re, OX, 2000)\n')
mf.write('Plane_Abgr_tab_li = geompy.MakePlane(Punkt_Abgr_li, OX, 2000)\n')
mf.write('\n')
mf.write('geompy.addToStudy( O, "O" )\n')
mf.write('geompy.addToStudy( OX, "OX" )\n')
mf.write('geompy.addToStudy( OY, "OY" )\n')
mf.write('geompy.addToStudy( OZ, "OZ" )\n')
mf.write('geompy.addToStudy( Punkt_Abgr_li, "Punkt_Abgr_li" )\n')
mf.write('geompy.addToStudy( Punkt_Abgr_re, "Punkt_Abgr_re" )\n')
mf.write('geompy.addToStudy( Plane_Abgr_tab_re, "Plane_Abgr_tab_re" )\n')
mf.write('geompy.addToStudy( Plane_Abgr_tab_li, "Plane_Abgr_tab_li" )\n')
mf.write('\n\n')
mf.write('#Erstellen der Punkte\n')
mf.write('\n')
mf.write('s=-lh\n')
mf.write('\n')
mf.write('punkteLi = []\n')
mf.write('punkteRe = []\n')
mf.write('schichten = []\n')
mf.write('Solid = []\n')
mf.write('\n')
mf.write('for i in range (0, al):\n')
mf.write('\n')
mf.write('#Einzelne Punkte erstellen\n')
mf.write('    s=s+lh\n')
mf.write('    punkteLi.append(geompy.MakeVertex(0, 0, s))\n')
mf.write('    Punkt_l_i = punkteLi[-1]\n')
mf.write('    geompy.addToStudy( Punkt_l_i, "Punkt_l_"+str(i+1) )\n')
mf.write('    punkteRe.append(geompy.MakeVertex(pl, pb, s+lh))\n')
mf.write('    Punkt_r_i = punkteRe[-1]\n')
mf.write('    geompy.addToStudy( Punkt_r_i, "Punkt_r_"+str(i+1) )\n')
mf.write('\n')
mf.write('#Einzelne Schichten erstellen\n')
mf.write('    schichten.append(geompy.MakeBoxTwoPnt(Punkt_l_i, Punkt_r_i))\n')
mf.write('    Schicht_i = schichten[-1]\n')
mf.write('    geompy.addToStudy( Schicht_i, "Schicht_"+str(i+1) )\n')
mf.write('    Solid.append("Solid_L"+str(i+1))\n')
mf.write('    Solid.append("Solid_M"+str(i+1))\n')
mf.write('    Solid.append("Solid_R"+str(i+1))\n')
mf.write('\n\n')
mf.write('#Gruppe aus Schichten erstellen    \n')
mf.write('Gruppe = geompy.MakeCompound(schichten)\n')
```

```
    mf.write('geompy.addToStudy( Gruppe, "Gruppe" )    \n')
    mf.write('\n\n')
    mf.write('#Partition aus Gruppe und Flaeche erstellen\n')
    mf.write('Probe = geompy.MakePartition([Gruppe], [Plane_Abgr_tab_re, Plane_Abgr_tab_li, ], [], [],
geompy.ShapeType["SOLID"], 0, [], 0)\n')
    mf.write('geompy.addToStudy( Probe, "Probe" )\n')
    mf.write('\n\n')
    mf.write('#Einzelne Volumenkoerper erstellen\n')
    mf.write('Solid = geompy.ExtractShapes(Probe, geompy.ShapeType["SOLID"], True)\n')
    mf.write('\n')
    mf.write('for i in range(0, al):\n')
    mf.write('    geompy.addToStudyInFather( Probe, Solid[0+i], "Solid_L"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Solid[al+i], "Solid_M"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Solid[al*2+i], "Solid_R"+str(i+1) )\n')
    mf.write('\n')
    mf.write('Solid_Mitte=[]\n')
    mf.write('Solid_Mitte = geompy.CreateGroup(Probe, geompy.ShapeType["SOLID"])\n')
    mf.write('geompy.UnionList(Solid_Mitte, Solid[al:al*2])\n')
    mf.write('geompy.addToStudyInFather( Probe, Solid_Mitte, "Solid_Mitte" )\n')
    mf.write('Solid_Rechts=[]\n')
    mf.write('Solid_Rechts = geompy.CreateGroup(Probe, geompy.ShapeType["SOLID"])\n')
    mf.write('geompy.UnionList(Solid_Rechts, Solid[al*2:al*3])\n')
    mf.write('geompy.addToStudyInFather( Probe, Solid_Rechts, "Solid_Rechts" )\n')
    mf.write('Solid_Links=[]\n')
    mf.write('Solid_Links = geompy.CreateGroup(Probe, geompy.ShapeType["SOLID"])\n')
    mf.write('geompy.UnionList(Solid_Links, Solid[0:al])\n')
    mf.write('geompy.addToStudyInFather( Probe, Solid_Links, "Solid_Links" )\n')
    mf.write('\n\n')
    mf.write('#Erstellen Edges\n')
    mf.write('Edge = []\n')
    mf.write('Edge = geompy.ExtractShapes(Probe, geompy.ShapeType["EDGE"], True)\n')
    mf.write('Edge_Hoch = geompy.MakeCompound(Edge[0:al])\n')
    mf.write('Edge_Breit = geompy.MakeCompound ([Edge[al]])\n')
    mf.write('geompy.addToStudyInFather( Probe, Edge_Hoch, "Edge_Hoch" )\n')
    mf.write('geompy.addToStudyInFather( Probe, Edge_Breit, "Edge_Breit" )\n')
    mf.write('\n\n')
    mf.write('#Einzelne Faces erstellen\n')
    mf.write('Face = []\n')
    mf.write('Face = geompy.ExtractShapes(Probe, geompy.ShapeType["FACE"], True)\n')
    mf.write('\n')
    mf.write('for i in range(0, al):\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*0+i], "Face_LL"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*2+i], "Face_LU"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*6+1+i], "Face_MU"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*10+2+i], "Face_RU"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*4+1+i], "Face_LM"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*8+2+i], "Face_RM"+str(i+1) )\n')
    mf.write('    geompy.addToStudyInFather( Probe, Face[al*12+3+i], "Face_RR"+str(i+1) )\n')
    mf.write('geompy.addToStudyInFather( Probe, Face[al*3], "Face_LU"+str(al+1) )\n')
    mf.write('geompy.addToStudyInFather( Probe, Face[al*7+1], "Face_MU"+str(al+1) )\n')
    mf.write('geompy.addToStudyInFather( Probe, Face[al*11+2], "Face_RU"+str(al+1) )\n')
    mf.write('\n\n')
    mf.write('#Gruppe fuer Knoten Erstellen\n')
```

```
    mf.write('Partition_LL = geompy.CreateGroup(Probe, geompy.ShapeType["FACE"])\n')
    mf.write('geompy.UnionList(Partition_LL, Face[al*0: al*1])\n')
    mf.write('geompy.addToStudyInFather( Probe, Partition_LL, "Partition_LL" )\n')
    mf.write('\n')
    mf.write('Partition_LM = geompy.CreateGroup(Probe, geompy.ShapeType["FACE"])\n')
    mf.write('geompy.UnionList(Partition_LM, Face[al*4+1: al*5+1])\n')
    mf.write('geompy.addToStudyInFather( Probe, Partition_LM, "Partition_LM" )\n')
    mf.write('\n')
    mf.write('Partition_RM = geompy.CreateGroup(Probe, geompy.ShapeType["FACE"])\n')
    mf.write('geompy.UnionList(Partition_RM, Face[al*8+2: al*9+2])\n')
    mf.write('geompy.addToStudyInFather( Probe, Partition_RM, "Partition_RM" )\n')
    mf.write('\n')
    mf.write('Partition_RR = geompy.CreateGroup(Probe, geompy.ShapeType["FACE"])\n')
    mf.write('geompy.UnionList(Partition_RR, Face[al*12+3: al*13+3])\n')
    mf.write('geompy.addToStudyInFather( Probe, Partition_RR, "Partition_RR" )\n')
    mf.write('\n\n\n')
    mf.write('###\n')
    mf.write('### SMESH component\n')
    mf.write('###\n')
    mf.write('\n\n')
    mf.write('import  SMESH, SALOMEDS\n')
    mf.write('from salome.smesh import smeshBuilder\n')
    mf.write('\n')
    mf.write('smesh = smeshBuilder.New(theStudy)\n')
    mf.write('Mesh_1 = smesh.Mesh(Probe)\n')
    mf.write('Regular_1D_1 = Mesh_1.Segment()\n')
    mf.write('Anzahl_Laenge = Regular_1D_1.NumberOfSegments(anl)\n')
    mf.write('Quadrangle_2D_1_1 = Mesh_1.Quadrangle(algo=smeshBuilder.QUADRANGLE)\n')
    mf.write('Hexa_3D_1 = Mesh_1.Hexahedron(algo=smeshBuilder.Hexa)\n')
    mf.write('Regular_1D_2 = Mesh_1.Segment(geom=Edge_Hoch)\n')
    mf.write('Anzahl_Hoehe = Regular_1D_2.NumberOfSegments(anh)\n')
    mf.write('Propagation_of_1D_Hyp = Regular_1D_2.Propagation()\n')
    mf.write('Regular_1D_3 = Mesh_1.Segment(geom=Edge_Breit)\n')
    mf.write('Anzahl_Netz_Breite = Regular_1D_3.NumberOfSegments(anb)\n')
    mf.write('Propagation_of_1D_Hyp_1 = Regular_1D_3.Propagation()\n')
    mf.write('Anzahl_Hoehe.SetNumberOfSegments( anh )\n')
    mf.write('isDone = Mesh_1.Compute()\n')
    mf.write('Sub_mesh_Hoehe = Regular_1D_2.GetSubMesh()\n')
    mf.write('Sub_mesh_Breite = Regular_1D_3.GetSubMesh()\n')
    mf.write('\n')
    mf.write('smesh.SetName(Mesh_1.GetMesh(), "Mesh_1")\n')
    mf.write('smesh.SetName(Anzahl_Laenge, "Anzahl_Laenge")\n')
    mf.write('smesh.SetName(Anzahl_Hoehe, "Anzahl_Hoehe")\n')
    mf.write('smesh.SetName(Propagation_of_1D_Hyp, "Propagation of 1D Hyp. on Opposite
Edges_2")\n')
    mf.write('smesh.SetName(Anzahl_Netz_Breite, "Anzahl_Netz_Breite")\n')
    mf.write('smesh.SetName(Propagation_of_1D_Hyp_1, "Propagation of 1D Hyp. on Opposite
Edges_1")\n')
    mf.write('smesh.SetName(Sub_mesh_Hoehe, "Sub_mesh_Hoehe")\n')
    mf.write('smesh.SetName(Sub_mesh_Breite, "Sub_mesh_Breite")\n')
    mf.write('\n\n')
    mf.write('#Solids in Mesh Gruppe uebertragen\n')
    mf.write('for i in range(0, al):\n')
```

```
    mf.write('\n')
    mf.write('    Solid[0+i] = Mesh_1.GroupOnGeom(Solid[0+i],"Solid_L"+str(i+1),SMESH.VOLUME)\n')
    mf.write('    Solid[al+i] = Mesh_1.GroupOnGeom(Solid[al+i],"Solid_M"+str(i+1),SMESH.VOLUME)\n')
    mf.write('    Solid[al*2+i] =
Mesh_1.GroupOnGeom(Solid[al*2+i],"Solid_R"+str(i+1),SMESH.VOLUME)\n')
    mf.write('Solid_Mitte = Mesh_1.GroupOnGeom(Solid_Mitte,"Solid_Mitte",SMESH.VOLUME)\n')
    mf.write('Solid_Rechts = Mesh_1.GroupOnGeom(Solid_Rechts,"Solid_Rechts",SMESH.VOLUME)\n')
    mf.write('Solid_Links = Mesh_1.GroupOnGeom(Solid_Links,"Solid_Links",SMESH.VOLUME)\n')
    mf.write('\n\n')
    mf.write('#Einzelne Lagen erstellen\n')
    mf.write('Lage=[]\n')
    mf.write('for i in range(0,al):\n')
    mf.write('    Lage.append( Mesh_1.GetMesh().UnionListOfGroups( [Solid[0+i], Solid[al+i],
Solid[2*al+i]], "Lage_"+str(i+1) ))\n')
    mf.write('    smesh.SetName(Lage[0+i], "Lage_"+str(1+i))\n')
    mf.write('\n\n')
    mf.write('#Knoten fuer die Einspannungen erstellen\n')
    mf.write('Knoten_LLL = Mesh_1.GroupOnGeom(Partition_LL,"Knoten_LLL",SMESH.NODE)\n')
    mf.write('smesh.SetName(Knoten_LLL, "Knoten_LLL")\n')
    mf.write('\n')
    mf.write('Knoten_LLM = Mesh_1.GroupOnGeom(Partition_LM,"Knoten_LLM",SMESH.NODE)\n')
    mf.write('smesh.SetName(Knoten_LLM, "Knoten_LLM")\n')
    mf.write('\n')
    mf.write('Knoten_LMM = Mesh_1.GroupOnGeom(Partition_RM,"Knoten_LMM",SMESH.NODE)\n')
    mf.write('smesh.SetName(Knoten_LMM, "Knoten_LMM")\n')
    mf.write('\n')
    mf.write('Knoten_RMM = Mesh_1.GroupOnGeom(Partition_RR,"Knoten_RMM",SMESH.NODE)\n')
    mf.write('smesh.SetName(Knoten_RMM, "Knoten_RMM")\n')
    mf.write('\n')
    mf.write('Knoten_1 = Mesh_1.CreateEmptyGroup( SMESH.NODE, "Knoten_1" )\n')
    mf.write('nbAdd = Knoten_1.Add( [ 1 ] )\n')
    mf.write('smesh.SetName(Knoten_1, "Knoten_1")\n')
    mf.write('\n')
    mf.write('Edge_1 = Mesh_1.GroupOnGeom(Edge_Breit,"Edge_1",SMESH.NODE)\n')
    mf.write('smesh.SetName(Edge_1, "Edge_1")\n')
    mf.write('\n')
    mf.write('#Elemente Mittig\n')
    mf.write('ElementS=[]\n')
    mf.write('for i in range(0, al):\n')
    mf.write('    ElementS.append( Mesh_1.GetMesh().UnionListOfGroups(
[Solid[al*1+i]],"ElementS"+str(i+1)))\n')
    mf.write('    smesh.SetName(ElementS[i], "ElementS"+str(i+1))\n')
    mf.write('\n\n')
    mf.write('#Export von Mesh im MED Format\n')
    mf.write('Mesh_1.ExportMED(r"'+inputDict['studyDir']+'/Case_1.med", 0 )\n')
    mf.write('\n\n')
    mf.write('if salome.sg.hasDesktop():\n')
    mf.write('    salome.sg.updateObjBrowser(True)\n')
    mf.write('\n')

def createAsterExportFile(inputDict):
  print(inputDict)
  with open(studyDir+'/Case_1.export', 'w') as ef:
```

```
        ef.write('P actions make_etude\n')
        ef.write('\n')
        ef.write('P memjob 1048576\n')
        ef.write('\n')
        ef.write('P memory_limit 10000.0\n')
        ef.write('\n')
        ef.write('P mode interactif\n')
        ef.write('\n')
        ef.write('P mpi_nbcpu 1\n')
        ef.write('\n')
        ef.write('P mpi_nbnoeud 1\n')
        ef.write('\n')
        ef.write('P time_limit 3600.0\n')
        ef.write('\n')
        ef.write('P tpsjob 16\n')
        ef.write('\n')
        ef.write('P version stable\n')
        ef.write('\n')
        ef.write('A memjeveux 128.0\n')
        ef.write('\n')
        ef.write('A tpmax 900.0\n')
        ef.write('\n')
        ef.write('F comm Case_1.comm D  1\n')
        ef.write('\n')
        ef.write('F libr Case_1.med D  20\n')
        ef.write('\n')
        ef.write('F libr ForcesL.med R  30\n')
        ef.write('\n')
        ef.write('F libr ForcesR.med R  31\n')
        ef.write('\n')
        for i in range(0, (inputDict['numberOfLayers'])):
            ef.write('F libr StressS'+str(i+1)+'.med R  '+str(i+60)+'\n')
            ef.write('\n')
#       ef.write('F libr Tabelle2.med R  11\n')
#       ef.write('\n')
        ef.write('F libr Results.med R  3\n')
        ef.write('\n')
        ef.write('F mess message.mess R  6\n')
        ef.write('\n')
        ef.write('\n')


def createAsterCommandFile(inputDict):
    oneAngle = False
    print(inputDict)
    with open(studyDir+'/Case_1.comm', 'w') as cf:
        cf.write('DEBUT()\n')
        cf.write('\n')
        if oneAngle:
            cf.write('L1Z = '+str(inputDict['Lage_1Z'])+' #Lage_1Z\n')
            cf.write('L2Z = '+str(inputDict['Lage_1Z'])+' #Lage_2Z\n')
            cf.write('L3Z = '+str(inputDict['Lage_1Z'])+' #Lage_3Z\n')
            cf.write('L4Z = '+str(inputDict['Lage_1Z'])+' #Lage_4Z\n')
```

```
    cf.write('L5Z = '+str(inputDict['Lage_1Z'])+' #Lage_5Z\n')
    cf.write('L6Z = '+str(inputDict['Lage_1Z'])+' #Lage_6Z\n')
    cf.write('L7Z = '+str(inputDict['Lage_1Z'])+' #Lage_7Z\n')
    cf.write('L8Z = '+str(inputDict['Lage_1Z'])+' #Lage_8Z\n')
    cf.write('L1Y = '+str(inputDict['Lage_1Y'])+' #Lage_1Y\n')
    cf.write('L2Y = '+str(inputDict['Lage_1Y'])+' #Lage_2Y\n')
    cf.write('L3Y = '+str(inputDict['Lage_1Y'])+' #Lage_3Y\n')
    cf.write('L4Y = '+str(inputDict['Lage_1Y'])+' #Lage_4Y\n')
    cf.write('L5Y = '+str(inputDict['Lage_1Y'])+' #Lage_5Y\n')
    cf.write('L6Y = '+str(inputDict['Lage_1Y'])+' #Lage_6Y\n')
    cf.write('L7Y = '+str(inputDict['Lage_1Y'])+' #Lage_7Y\n')
    cf.write('L8Y = '+str(inputDict['Lage_1Y'])+' #Lage_8Y\n')
    cf.write('L1X = '+str(inputDict['Lage_1X'])+' #Lage_1X\n')
    cf.write('L2X = '+str(inputDict['Lage_1X'])+' #Lage_2X\n')
    cf.write('L3X = '+str(inputDict['Lage_1X'])+' #Lage_3X\n')
    cf.write('L4X = '+str(inputDict['Lage_1X'])+' #Lage_4X\n')
    cf.write('L5X = '+str(inputDict['Lage_1X'])+' #Lage_5X\n')
    cf.write('L6X = '+str(inputDict['Lage_1X'])+' #Lage_6X\n')
    cf.write('L7X = '+str(inputDict['Lage_1X'])+' #Lage_7X\n')
    cf.write('L8X = '+str(inputDict['Lage_1X'])+' #Lage_8X\n')
else:
    cf.write('L1Z = '+str(inputDict['Lage_1Z'])+' #Lage_1Z\n')
    cf.write('L2Z = '+str(inputDict['Lage_2Z'])+' #Lage_2Z\n')
    cf.write('L3Z = '+str(inputDict['Lage_3Z'])+' #Lage_3Z\n')
    cf.write('L4Z = '+str(inputDict['Lage_4Z'])+' #Lage_4Z\n')
    cf.write('L5Z = '+str(inputDict['Lage_5Z'])+' #Lage_5Z\n')
    cf.write('L6Z = '+str(inputDict['Lage_6Z'])+' #Lage_6Z\n')
    cf.write('L7Z = '+str(inputDict['Lage_7Z'])+' #Lage_7Z\n')
    cf.write('L8Z = '+str(inputDict['Lage_8Z'])+' #Lage_8Z\n')
    cf.write('L1Y = '+str(inputDict['Lage_1Y'])+' #Lage_1Y\n')
    cf.write('L2Y = '+str(inputDict['Lage_2Y'])+' #Lage_2Y\n')
    cf.write('L3Y = '+str(inputDict['Lage_3Y'])+' #Lage_3Y\n')
    cf.write('L4Y = '+str(inputDict['Lage_4Y'])+' #Lage_4Y\n')
    cf.write('L5Y = '+str(inputDict['Lage_5Y'])+' #Lage_5Y\n')
    cf.write('L6Y = '+str(inputDict['Lage_6Y'])+' #Lage_6Y\n')
    cf.write('L7Y = '+str(inputDict['Lage_7Y'])+' #Lage_7Y\n')
    cf.write('L8Y = '+str(inputDict['Lage_8Y'])+' #Lage_8Y\n')
    cf.write('L1X = '+str(inputDict['Lage_1X'])+' #Lage_1X\n')
    cf.write('L2X = '+str(inputDict['Lage_2X'])+' #Lage_2X\n')
    cf.write('L3X = '+str(inputDict['Lage_3X'])+' #Lage_3X\n')
    cf.write('L4X = '+str(inputDict['Lage_4X'])+' #Lage_4X\n')
    cf.write('L5X = '+str(inputDict['Lage_5X'])+' #Lage_5X\n')
    cf.write('L6X = '+str(inputDict['Lage_6X'])+' #Lage_6X\n')
    cf.write('L7X = '+str(inputDict['Lage_7X'])+' #Lage_7X\n')
    cf.write('L8X = '+str(inputDict['Lage_8X'])+' #Lage_8X\n')
cf.write('\n')
cf.write('Mesh = LIRE_MAILLAGE(FORMAT="MED", UNITE=20)\n')
cf.write('\n')
cf.write('M_Alles = AFFE_MODELE(\n')
cf.write('    AFFE=_F(MODELISATION=("3D", ), PHENOMENE="MECANIQUE", TOUT="OUI"),\n')
cf.write('    MAILLAGE=Mesh)\n')
cf.write('\n')
cf.write('EinzLage = AFFE_CARA_ELEM(\n')
```

```
      cf.write('    MASSIF=(\n')
      for i in range(0, ((inputDict['numberOfLayers'])-1)):
         cf.write('       _F(ANGL_REP=(L'+str(i+1)+'Z, L'+str(i+1)+'Y, L'+str(i+1)+'X),
GROUP_MA=("Lage_'+str(i+1)+'")),\n')
      cf.write('       _F(ANGL_REP=(L'+str((inputDict['numberOfLayers']))+'Z,
L'+str((inputDict['numberOfLayers']))+'Y, L'+str((inputDict['numberOfLayers']))+'X),
GROUP_MA=("Lage_'+str((inputDict['numberOfLayers']))+'"))), \n')
      cf.write('    MODELE=M_Alles)\n')
      cf.write('\n')
      cf.write('MoD = DEFI_MATERIAU(\n')
      cf.write('    ELAS_ORTH=_F(\n')
      cf.write('       E_L='+str(inputDict['E_L'])+' ,\n')
      cf.write('       E_N='+str(inputDict['E_N'])+' ,\n')
      cf.write('       E_T='+str(inputDict['E_N'])+' ,\n')
      cf.write('       G_LN='+str(inputDict['G_LN'])+' ,\n')
      cf.write('       G_LT='+str(inputDict['G_LT'])+' ,\n')
      cf.write('       G_TN='+str(inputDict['G_TN'])+' ,\n')
      cf.write('       NU_LN='+str(inputDict['NU_LN'])+' ,\n')
      cf.write('       NU_LT='+str(inputDict['NU_LT'])+' ,\n')
      cf.write('       NU_TN='+str(inputDict['NU_TN'])+' ))\n')
      cf.write('\n')
      cf.write('COMP_oD = AFFE_MATERIAU(AFFE=_F(MATER=(MoD, ), TOUT="OUI"),
MODELE=M_Alles)\n')
      cf.write('\n')
      cf.write('Fix_Li = AFFE_CHAR_MECA(\n')
      cf.write('    DDL_IMPO=(\n')
      cf.write('    _F(DX=0.0, GROUP_NO=("Knoten_LLL", )),\n')
      cf.write('    _F(DY=0.0, GROUP_NO=("Knoten_1", )),\n')
      cf.write('    _F(DZ=0.0, GROUP_NO=("Edge_1", ))),\n')
      cf.write('MODELE=M_Alles)\n')
      cf.write('\n')
      cf.write('Displ = AFFE_CHAR_MECA(\n')
      cf.write('    DDL_IMPO=_F(DX='+str(inputDict['DISPL'])+' , GROUP_NO=("Knoten_RMM", )),
MODELE=M_Alles)\n')
      cf.write('\n')
      cf.write('Res_oD = MECA_STATIQUE(\n')
      cf.write('    CARA_ELEM=EinzLage,\n')
      cf.write('    CHAM_MATER=COMP_oD,\n')
      cf.write('    EXCIT=(_F(CHARGE=Displ), _F(CHARGE=Fix_Li)),\n')
      cf.write('    MODELE=M_Alles)\n')
      cf.write('\n')
      cf.write('Res_oD = CALC_CHAMP(\n')
      cf.write('    reuse=Res_oD,\n')
      cf.write('    CONTRAINTE=("SIGM_ELNO", ),\n')
#       cf.write('    CRITERES=("SIEQ_ELGA", "SIEQ_NOEU"),\n')
#       cf.write('    DEFORMATION=("DEGE_ELNO", "EPSI_ELNO"),\n')
      cf.write('    FORCE=("FORC_NODA", "REAC_NODA"),\n')
      cf.write('    RESULTAT=Res_oD)\n')
      cf.write('\n')
      cf.write('ForcesL = CREA_TABLE(\n')
      cf.write('    RESU=_F(\n')
      cf.write('       GROUP_NO=("Knoten_LLL",),\n')
      cf.write('       NOM_CHAM="REAC_NODA",\n')
```

```
    cf.write('       RESULTAT=Res_oD,\n')
    cf.write('       TOUT_CMP="OUI"),\n')
    cf.write('    TITRE="REAC_NODA")\n')
    cf.write('\n')
    cf.write('ForcesR = CREA_TABLE(\n')
    cf.write('    RESU=_F(\n')
    cf.write('       GROUP_NO=("Knoten_RMM",),\n')
    cf.write('       NOM_CHAM="FORC_NODA",\n')
    cf.write('       RESULTAT=Res_oD,\n')
    cf.write('       TOUT_CMP="OUI"),\n')
    cf.write('    TITRE="FORC_NODA")\n')
    cf.write('\n')
    cf.write('IMPR_TABLE(FORMAT="ASTER", \n')
    cf.write('    NOM_PARA=("DX", "NOEUD",), \n') #Ausgabe nur Kraefte in "DX" Richtung
    cf.write('    TABLE=ForcesL, UNITE=30)\n')
    cf.write('\n')
    cf.write('IMPR_TABLE(FORMAT="ASTER", \n')
    cf.write('    NOM_PARA=("DX",), \n') #Ausgabe nur Kraefte in "DX" Richtung
    cf.write('    TABLE=ForcesR, UNITE=31)\n')
    cf.write('\n')
    for i in range(0, (inputDict['numberOfLayers'])):
      cf.write('StressS'+str(i+1)+'= CREA_TABLE(\n')
      cf.write('    RESU=_F(\n')
      cf.write('       GROUP_MA="ElementS'+str(i+1)+'" ,\n')
      cf.write('       NOM_CHAM="SIGM_ELNO",\n')
      cf.write('       RESULTAT=Res_oD,\n')
      cf.write('       TOUT_CMP="OUI"),\n')
      cf.write('    TITRE="SIGM_NOEU")\n')
      cf.write('\n')
    for i in range(0, (inputDict['numberOfLayers'])):
      cf.write('IMPR_TABLE(FORMAT="ASTER", TABLE=StressS'+str(i+1)+', UNITE='+str(i+60)+')\n')
      #Tabelle Ausgabe
      #cf.write('    NOM_PARA=("NOM_CHAM", "NOEUD", "COOR_X", "COOR_Y", "COOR_Z", \n')
      #cf.write('       "SIXX", "SIYY", "SIZZ", "SIXY", "SIXZ", "SIYZ", ),\n')
      cf.write('\n')
    cf.write('IMPR_RESU(FORMAT="MED", RESU=_F(RESULTAT=Res_oD), UNITE=3)\n')
    cf.write('\n')
    cf.write('FIN()\n')
    cf.write('\n')

def evaluateStiffness(parametersDict):
  with open(studyDir+'/ForcesR.med','r') as rf:
    reactionForces = np.loadtxt(rf, comments='#', skiprows=8, usecols=(0,))

    stiffness = np.sum(reactionForces)/(parametersDict['DISPL']*parametersDict['laminatHeight'])

  return stiffness

def evaluateFirstPlyFailure(parametersDict):
  for layerNumber in range(parametersDict['numberOfLayers']):
    with open(studyDir+'/StressS'+str(layerNumber+1)+'.med','r') as rf:
      layerStresses = np.loadtxt(rf, comments='#', skiprows=8, usecols=(10,11,12,15,14,13))
```

```
        layerAngle = parametersDict['Lage_'+str(layerNumber+1)+'Z']
        layerTrafo = Transformation()
        layerTrafo.setRotationByAxisAndAngle([0., 0., 1.], layerAngle)

        rotationMatrix = layerTrafo.getRotation().T
        layerStressTrafo = smetanaUtils.getTransformationMatrixFromMatrix(rotationMatrix, 'epsilon')[1].T

        rotatedLayerStresses = [np.dot(layerStressTrafo, stresses) for stresses in layerStresses]
#       for stresses in layerStresses:
#           rotatedLayerStresses.append(((np.dot(layerStressTrafo, stresses)).T))
#
#       rotatedLayerStresses.extend([np.dot(layerStressTrafo, stresses) for stresses in layerStresses])

        strength = [2610., 1450., 55., 285., 105., 64.5, 250., 68., 68.]
        failEval = MCDFibreMaxStress().getFailureResultsDict(rotatedLayerStresses,
[strength]*len(rotatedLayerStresses))

    return np.max(failEval['criticalFailureIndex'])

def scaleParametersToBounds(parametersList, boundsDict, sample):
    scaledSample = []
    for parameterName, parameterValue in zip(parametersList,sample):
        scaledParameter =
(boundsDict[parameterName][0]+parameterValue*(boundsDict[parameterName][1]-
boundsDict[parameterName][0]))
        scaledSample.append(scaledParameter)

    retPoint = ot.Point(scaledSample)
    return retPoint

if __name__ == '__main__':
    performMorrisScreening = True
    evaluateResults = True

    #parmUniformDist = ot.Uniform(87.,93.)
    #parmUniformDist.setName('Gleichverteilung von ParameterXY')
    #parmUniformDist.setDescription(['ParameterXY'])

    if performMorrisScreening:
        parmNormalDist1 = ot.Uniform(0.,1.)
        parmNormalDist1.setName('Normalverteilung von E_L')
        parmNormalDist1.setDescription(['E_L'])

        parmNormalDist2 = ot.Uniform(0.,1.)
        parmNormalDist2.setName('Normalverteilung von E_N')
        parmNormalDist2.setDescription(['E_N' ])

        parmNormalDist3 = ot.Uniform(0.,1.)
        parmNormalDist3.setName('Normalverteilung von G_LN')
        parmNormalDist3.setDescription(['G_LN' ])

        parmNormalDist4 = ot.Uniform(0.,1.)
        parmNormalDist4.setName('Normalverteilung von NU_LN')
```

```
parmNormalDist4.setDescription(['NU_LN' ])

#Lagenorientierung
parmNormalDist5 = ot.Uniform(0.,1.)
parmNormalDist5.setName('Normalverteilung von Lage_1Z')
parmNormalDist5.setDescription(['Lage_1Z' ])

parmNormalDist6 = ot.Uniform(0.,1.)
parmNormalDist6.setName('Normalverteilung von Lage_2Z')
parmNormalDist6.setDescription(['Lage_2Z' ])

parmNormalDist7 = ot.Uniform(0.,1.)
parmNormalDist7.setName('Normalverteilung von Lage_3Z')
parmNormalDist7.setDescription(['Lage_3Z' ])

parmNormalDist8 = ot.Uniform(0.,1.)
parmNormalDist8.setName('Normalverteilung von Lage_4Z')
parmNormalDist8.setDescription(['Lage_4Z' ])

parmNormalDist9 = ot.Uniform(0.,1.)
parmNormalDist9.setName('Normalverteilung von Lage_5Z')
parmNormalDist9.setDescription(['Lage_5Z' ])

parmNormalDist10 = ot.Uniform(0.,1.)
parmNormalDist10.setName('Normalverteilung von Lage_6Z')
parmNormalDist10.setDescription(['Lage_6Z' ])

parmNormalDist11 = ot.Uniform(0.,1.)
parmNormalDist11.setName('Normalverteilung von Lage_7Z')
parmNormalDist11.setDescription(['Lage_7Z' ])

parmNormalDist12 = ot.Uniform(0.,1.)
parmNormalDist12.setName('Normalverteilung von Lage_8Z')
parmNormalDist12.setDescription(['Lage_8Z' ])

#Lagendicke
parmNormalDist13 = ot.Uniform(0.,1.)
parmNormalDist13.setName('Normalverteilung von Laminatdicke')
parmNormalDist13.setDescription(['laminatHeight' ])

#Probengeometrie
parmNormalDist14 = ot.Uniform(0.,1.)
parmNormalDist14.setName('Normalverteilung von Breite')
parmNormalDist14.setDescription(['specimenWidth' ])

parmNormalDist15 = ot.Uniform(0.,1.)
parmNormalDist15.setName('Normalverteilung von Laenge')
parmNormalDist15.setDescription(['specimenLength' ])

#Verschiebung
parmNormalDist16 = ot.Uniform(0.,1.)
parmNormalDist16.setName('Normalverteilung von Verschiebung')
parmNormalDist16.setDescription(['DISPL' ])
```

```
    numberVariables = 16
    independentCopula = ot.IndependentCopula(numberVariables)
    independentCopula.setName('Independent copula')

    jointDistribution = ot.ComposedDistribution([parmNormalDist1, parmNormalDist2,
parmNormalDist3, parmNormalDist4, parmNormalDist5,
                                parmNormalDist6, parmNormalDist7, parmNormalDist8,
parmNormalDist9, parmNormalDist10,
                                parmNormalDist11, parmNormalDist12, parmNormalDist13,
parmNormalDist14, parmNormalDist15,
                                parmNormalDist16, ], independentCopula)

    boundsDict = {'DISPL':(0.095,0.105), 'specimenWidth':(21.8,22.2), 'specimenLength':(21.5,22.5),
'laminatHeight':(1.8,2,2),
            'Lage_1Z':(-3.,3.), 'Lage_2Z':(-3.,3.), 'Lage_3Z':(-3.,3.), 'Lage_4Z':(-3.,3.), 'Lage_5Z':(-3.,3.),
'Lage_6Z':(-3.,3.), 'Lage_7Z':(-3.,3.), 'Lage_8Z':(-3.,3.),
            'E_L':(146700.,179300.), 'E_N':(7650.,9350.), 'G_LN':(3780.,4620.), 'NU_LN':(0.315,0.385) }


    morrisInterval = ot.Interval([boundsDict[parmName][0] for parmName in
jointDistribution.getDescription()],
                    [boundsDict[parmName][1] for parmName in jointDistribution.getDescription()],
                    [True]*numberVariables, [True]*numberVariables)

    morrisInterval = ot.Interval(numberVariables)

  else:
    #MaterialParameter
    parmNormalDist1 = ot.Normal(163000.,16300.)
    parmNormalDist1.setName('Normalverteilung von E_L')
    parmNormalDist1.setDescription(['E_L'])

    parmNormalDist2 = ot.Normal(8500.,850.)
    parmNormalDist2.setName('Normalverteilung von E_N')
    parmNormalDist2.setDescription(['E_N' ])

    parmNormalDist3 = ot.Normal(4200.,420.)
    parmNormalDist3.setName('Normalverteilung von G_LN')
    parmNormalDist3.setDescription(['G_LN' ])

    parmNormalDist4 = ot.Normal(0.35,0.035)
    parmNormalDist4.setName('Normalverteilung von NU_LN')
    parmNormalDist4.setDescription(['NU_LN' ])

    #Lagenorientierung
    parmNormalDist5 = ot.Normal(0.,3.)
    parmNormalDist5.setName('Normalverteilung von Lage_1Z')
    parmNormalDist5.setDescription(['Lage_1Z' ])

    #Lagendicke
    parmNormalDist13 = ot.Normal(2.,0.2)
    parmNormalDist13.setName('Normalverteilung von Laminatdicke')
```

```
    parmNormalDist13.setDescription(['laminatHeight' ])

    #Verschiebung
    parmNormalDist15 = ot.Normal(-0.1,0.005)
    parmNormalDist15.setName('Normalverteilung von Verschiebung')
    parmNormalDist15.setDescription(['DISPL' ])

    numberVariables = 7
    independentCopula = ot.IndependentCopula(numberVariables)
    independentCopula.setName('Independent copula')

    jointDistribution = ot.ComposedDistribution([parmNormalDist1, parmNormalDist2,
parmNormalDist3, parmNormalDist4, parmNormalDist5,
                                parmNormalDist13, parmNormalDist15, ], independentCopula)

  jointDistribution.setName('jointDistribution')

  if evaluateResults:
    outputDesign =
ot.Sample().ImportFromCSVFile("D:/Projekte/SalomeMecaAndCodeAster/tmp/outputDesign.csv",',')

    if performMorrisScreening:
      # Evaluate Elementary effects (ee)
      ee = otmorris.Morris(outputDesign[:,:-2], outputDesign[:,-2:-1], morrisInterval)
      # Compute mu/sigma
      mean = ee.getMeanAbsoluteElementaryEffects()
      sigma = ee.getStandardDeviationElementaryEffects()
      fig = PlotEE(ee, title="Elementary Effects using LHS")
      fig.show()
    else:

      # Compute first order indices using the Saltelli estimator
      sensitivityAnalysis = ot.SaltelliSensitivityAlgorithm(outputDesign[:,:-2], outputDesign[:,-2:-1], size)
      first_order = sensitivityAnalysis.getFirstOrderIndices()
      print(first_order)

      # Draw indices
      graph = sensitivityAnalysis.draw()
      View(graph)

      # second order indices
      second_order = sensitivityAnalysis.getSecondOrderIndices()
      for i in range(numberVariables):
        for j in range(i):
          print('2nd order indice %d,%d=%g' % (i,j,second_order[i,j]))

  else:
    try:
      os.makedirs("D:/Projekte/SalomeMecaAndCodeAster/tmp/DruckversuchParameterstudie")
    except FileExistsError:
      pass

    if performMorrisScreening:
```

```
        size = 10000
        r = 15
        # Define experiments in [0,1]^2
        # Generate an LHS design
        # should be centered so randomShift=False
        lhs_experiment = ot.LHSExperiment(jointDistribution, size, True, False)
        lhsDesign = lhs_experiment.generate()
        experiment = otmorris.MorrisExperimentLHS(lhsDesign, r)
        inputDesign = experiment.generate()
        inputDesign.setDescription(jointDistribution.getDescription())

    else:
        size = 500
        inputDesign = ot.SobolIndicesExperiment(jointDistribution, size, True).generate()


inputDesign.exportToCSVFile("D:/Projekte/SalomeMecaAndCodeAster/tmp/DruckversuchParameterstu
die/inputDesign.csv", ',')

    outputDesign = []
    outSample = ot.Sample()
    offset = 0
    for it,sample in enumerate(inputDesign[offset:]):
        actualSample = sample

        parametersDict = {'a':10., 'name':'Test',
                'specimenWidth':32., 'specimenLength':32., 'tabLengthLeft':10., 'tabLengthRight':10.,
                'layerHeight':0.25, 'numberOfLayers':int(8),
                'numberMeshWidth':int(3), 'numberMeshHeight':int(1), 'numberMeshLength':int(2),
                'Lage_1Z':0.0, 'Lage_2Z':0.0,  'Lage_3Z':90.0, 'Lage_4Z':90.0, 'Lage_5Z':90.0,
'Lage_6Z':90.0, 'Lage_7Z':0.0, 'Lage_8Z':0.0,
                'Lage_1Y':0.0, 'Lage_2Y':0.0,  'Lage_3Y':0.0, 'Lage_4Y':0.0, 'Lage_5Y':0.0, 'Lage_6Y':0.0,
'Lage_7Y':0.0, 'Lage_8Y':0.0,
                'Lage_1X':0.0, 'Lage_2X':0.0, 'Lage_3X':0.0, 'Lage_4X':0.0, 'Lage_5X':0.0, 'Lage_6X':0.0,
'Lage_7X':0.0, 'Lage_8X':0.0,
                'E_L':163000.0, 'E_N':8500.0,
                'G_LN':4200.0, 'G_LT':4200.0, 'G_TN':3360.0,
                'NU_LN':0.35, 'NU_LT':0.35, 'NU_TN':0.26,
                'DISPL':-0.1 }


        if performMorrisScreening:
            actualSample = scaleParametersToBounds(inputDesign.getDescription(), boundsDict, sample)

        parametersDict.update(dict(zip(inputDesign.getDescription(),actualSample)))

        studyDir =
"D:/Projekte/SalomeMecaAndCodeAster/tmp/DruckversuchParameterstudie/Druckversuch"+str(it)
        try:
            os.makedirs(studyDir)
        except:
            pass
```

```
        parametersDict['studyDir'] = studyDir

        createRunSalomeBatch(studyDir, 'GeometrieVernetzung.py')
        createRunAsterBatch(studyDir, 'Case_1.export')
        createSalomeModelScript(parametersDict)
        callSalome(studyDir)
        createAsterExportFile(parametersDict)
        createAsterCommandFile(parametersDict)
        callAster(studyDir)

        #result evaluation
        outputSample = ot.Point(sample)

        stiffness = evaluateStiffness(parametersDict)
        outputSample.add(stiffness)

        matEffort = evaluateFirstPlyFailure(parametersDict)
        outputSample.add(matEffort)

        outputDesign.append(outputSample)
        tmpSample = ot.Sample(outputDesign)
        tmpSample.setDescription(list(inputDesign.getDescription())+['Stiffness', 'MaterialEffort'])

    outputDesign = ot.Sample(outputDesign)
    outputDesign.setDescription(list(inputDesign.getDescription())+['Stiffness', 'MaterialEffort'])

outputDesign.exportToCSVFile("D:/Projekte/SalomeMecaAndCodeAster/tmp/DruckversuchParameterst
udie/outputDesign.csv", ',')

#Konvergenzstudie

#   for (meshWidth, meshLength, meshHeight) in [ (3,1,2)]:
#
#       parametersDict['numberMeshWidth'] = meshWidth
#       parametersDict['numberMeshHeight'] = meshHeight
#       parametersDict['numberMeshLength'] = meshLength
#
#       studyDir = "D:/FEM-Tools/code-
aster/WorkSpace/DruckversuchParameterstudie/Druckversuch"+str(meshWidth)+'x'+str(meshLength)+'
x'+str(meshHeight)
#
#       os.makedirs(studyDir)
#
#       parametersDict['studyDir'] = studyDir

#       createRunSalomeBatch(studyDir, 'GeometrieVernetzung.py')
#       createRunAsterBatch(studyDir, 'Case_1.export')
#       createSalomeModelScript(parametersDict)
#       callSalome(studyDir)
#       createAsterExportFile(parametersDict)
#       createAsterCommandFile(parametersDict)
#       callAster(studyDir)
```

Anhang B: Selbständigkeitserklärung

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: Löwen

Vorname: Peter

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

Umgang mit fertigungsbedingten Unsicherheiten in der Analyse und Bewertung von Faserverbundstrukturen im Kontext des Flugzeuggesamtentwurfs

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 -bitte auswählen- ist erfolgt durch:

Braunschweig	03.12.2018	
Ort	Datum	Unterschrift im Original