

ENTWICKLUNG EINES MATLAB-TOOLS ZUR GENERIERUNG ENDKONTURNAHER VOXEL-NETZE FÜR TEXTILE HALBZEUGE

Development of a Matlab-tool to produce near net-shape
Voxel structures of textile-semi-finished products

Bachelorarbeit von **Christian Dreßler**

durchgeführt am

INSTITUT FÜR FLUGZEUGBAU
UNIVERSITÄT STUTTGART

Im Zeitraum von 05.04.16 bis 04.10.16

AUFGABENSTELLUNG

IFB Institut für Flugzeugbau · Universität Stuttgart

Leiter: Professor Dr.-Ing. Peter Middendorf

cand. aer. Christian Dressler
Böblinger Straße 27a
70178 Stuttgart
Matr.-Nr.: 2785103

Postadresse: D - 70550 Stuttgart
Hausadresse: Pfaffenwaldring 31
D - 70569 Stuttgart
Telefon: 49/(0)711/685-62770
Telefax: 49/(0)711/685-62065

Bachelorarbeit

- **Thema: Entwicklung eines Matlab-Tools zur Generierung endkonturnaher Voxel-Netze für textile Halbzeuge**

Kurzbeschreibung:

Die Serienproduktion von Faserverbundbauteilen im Automobilbau erfordert kurze Prozesszeiten und eine reproduzierbare Qualität. Diese beiden Anforderungen können bisher nur mittels RTM-Prozess realisiert werden. Beim RTM-Prozess (resin transfer molding) werden die trockenen Faserhalbzeuge in eine Form gelegt und anschließend wird das Injektionsharz mit Überdruck in die geschlossene Form gespritzt. Nach dem Aushärten des Bauteils kann dieses dann der Form entnommen und weiter verarbeitet werden. Um nicht infiltrierte Stellen im Bauteil zu vermeiden sind eine genaue Kenntnis der Permeabilitäten (Durchdringbarkeit) der Halbzeuge und eine Simulation des Füllvorgangs erforderlich. Mesoskopische Simulationen bieten den Vorteil realitätsnaher Faserarchitekturen, bringen aber z.T. enorme Rechenzeiten mit sich.

- Mesoskopische Abbildungen textiler Einheitszellen werden derzeit mit Hilfe von Softwaretools wie WiseTex und TexGen generiert. Die Anwendung auf endkonturnahen (drapierte, geflochtene) Bauteil ist nicht zweckmäßig, da die Rechenzeit zu groß wird, bzw. nicht möglich. Ein Voxel-Rechennetz hätte den Vorteil eines strukturierten Netzes, welches für die klassische FEM-Auslegung sowie für die CFD-Strömungssimulation deutlich schnellere Rechenzeiten erreichen könnte. Abweichungen in der Geometrie müssten analysiert und abgeschätzt und in Bezug zur schnelleren Rechenzeit gesetzt werden.

Hauptziel dieser Arbeit ist daher die Entwicklung eines MATLAB-Tools zur Generierung endkonturnaher Voxel-Netze für textile Halbzeuge. Hierbei sollen verschiedene Textilien aufgebaut und hinsichtlich Weiterverwendung getestet werden.

Arbeitspunkte:

- Literaturrecherche
- Erarbeitung Problemstellung
- Programmierung des MATLAB-Tools
- Test der Beispielgeometrien
- Analyse und Vergleich
- Dokumentation

Arbeit ausgegeben am: 05.04.16

Arbeit abgegeben am: 04.10.16

Betreuer: Dipl.-Ing. Jörg Dittmann (IFB), Dipl.-Ing. Patrick Böhler (IFB)

Stuttgart, 05.04.16

Prof. Dr.-Ing. Peter Middendorf

KURZFASSUNG

Die vorliegende Bachelorarbeit beschreibt eine Methode und deren programmierte Umsetzung zur Erzeugung realitätsnaher, virtueller Abbildungen der Fasern von Faserverbundbauteilen bei vergleichsweise geringem Rechenaufwand. Hierfür werden von bereits als zweidimensionale Flächen vorliegenden Faserverläufen zunächst eine geeignete Anzahl an einzelnen dreidimensionalen Körpern abgeleitet. Diese voneinander unabhängigen Objekte werden im weiteren Verlauf entsprechend den Eingabedaten miteinander verbunden. Durch die Weiterentwicklung der Faserbänder zu dreidimensionalen Objekten kommt es zu Kollisionen zwischen den einzelnen Elementen. Diese werden durch eine einfache Umformvorschrift weitestgehend beseitigt.

Um den Anforderungen an die darauffolgende Harzinjektionssimulation gerecht zu werden, wird das im ersten Schritt erzeugte Fasermodell kompatibel erweitert. Hinzugefügt wird eine Abbildung des Raumes, in welchem die Fasern eingebettet und von mit der Matrix umhüllt werden. Abschließend werden für das Modell die Ein- und Auslässe sowie die übrigen Randflächen aller Elemente für die Injektionssimulation definiert.

Der Programmcode liegt in der Programmiersprache Matlab vor. Die verwendete Funktionsbibliothek entspricht der Version 2015b. Das beiliegende Programm basiert auf einem gleichförmigen Gitter in dem alle Elemente als Würfel, sogenannte Voxel, diskretisiert werden. Durch diese Methode können alle vorkommenden Geometrien, innerhalb einer im Vergleich zu bisherigen Methoden kurzen Rechenzeit, entkennbar erstellt werden. Die wirkenden Kräfte und Momente werden im Voxelmodell nicht abgebildet.

Der im Rahmen der Arbeit entstandene Programmcode ist laufzeitoptimiert. Er enthält eine minimale Anzahl an impliziten Suchfunktionen. Um die zukünftige Wart- und Erweiterbarkeit auch durch weitere Autoren zu ermöglichen, entspricht der Aufbau der vorliegenden Ausarbeitung zu weiten Teilen dem Programmaufbau.

ABSTRACT

This bachelor thesis describes a method to produce realistic geometrical models of RTM-preforms with low computational resources to simulate the permeability of the provided unit-cell. The method is based on a structured voxel-mesh. The data-model is not taking any stresses into effect. Instead, the algorithms are solely based on experimentally gained rules. While limiting the use cases, this method decreases the required computing power to create geometric models of fiber-composite-parts substantially.

The required user interaction is limited to choosing the composite injection direction, the size of the cubic voxels and to enter the average yarn height. The textile input data consist of 2D shell-models of the yarn while the cavity data describe the cavity volume.

The yarns are created by sweeping an elliptical cross section along a pre-defined path along a shell and putting the shells together afterwards.

Unlike other voxel-based preprocessors this program can deform the yarns due to collisions with other yarns or fabrics.

Additionally, the empty cavity is filled and inlets and outlets are defined.

INHALTSVERZEICHNIS

Aufgabenstellung	I
Kurzfassung	II
Abstract	III
Inhaltsverzeichnis	IV
Nomenklatur	VI
Abbildungsverzeichnis	VIII
Tabellenverzeichnis	X
Untersuchung	1
1 Einleitung	1
2 Stand der Technik	4
2.1 Einordnung in den Workflow	4
2.2 Vorhandene Software	4
2.2.1 Wisetex	5
2.2.2 TexGen	9
2.3 Modellierung	14
2.4 Voxelerzeugung	16
2.5 QuerschnittsVerformung	17
3 Untersuchung	18
3.1 Generelles	18
3.1.2 Prinzipien	20
3.2 Verarbeitung der Faserdaten	20
3.2.2 Extrudieren	27
3.2.3 Erstellung der Übergänge	32
3.2.4 Kollisionsbedingtes Umformen	36
3.3 Verarbeitung der Kavitätsdaten	44
3.4 Ausgabefunktion	48
3.5 Optimierungsfunktionen	52
4 Zusammenfassung und Ausblick	56
Literaturverzeichnis	i
Anhang	iii
1 Messprotokolle	iii
1.1 Nutzen des Geordneten Gitters	vii
1.2 Indexierung	viii
1.3 Vernüpfung der Voxel-ID mit der Zeile	ix
1.4 Parallelisierung	ix
2 Funktionsdiagramme	x
2.1 F00_Main	x

2.2	F01_Einstellungen	xi
2.3	F02_Rovinge	xii
2.4	F02_1_Rovinge_trennen	xiii
2.5	F02_1_2_0_Zuordnung	xiv
2.6	F02_1_3_letzte_Flaeche.....	xv
2.7	F02_2_0_Uebergaenge.....	xvi
2.8	F02_2_2_Spiegelung.....	xvii
2.9	F02_3_0_Umformen	xviii
2.10	F02_3_1_0_Kollisionskoerper	xix
2.11	F02_3_2_0_Umformen_senkrecht.....	xx
2.12	F02_3_2_1_0_Spiegelfunktion.....	xxi
2.13	F02_3_2_1_3_0_Schmittflaeche_ermitteln.....	xxii
2.14	F02_3_3_0_Umformen_waagrecht.....	xxiii
2.15	F02_3_3_1_0_Verschiebungsfunktion.....	xxiv
2.16	F02_3_3_1_2_0_Verschiebungsfunk._Scheibe.....	xxv
2.17	F03_0_Kavitaet	xxvi
2.18	F03_2_0_Kavitaet_fuellen.....	xxvii
2.19	F03_3_0_InOutlets	xxviii
2.20	F04_0_Ausgabe	xxix
2.21	F04_1_0_Solids_generieren	xxx
2.22	F04_2_0_Shells_generieren.....	xxx
2.23	F04_2_2_1_Shells_filtern	xxxi
2.24	F04_3_Knoten_generieren.....	xxxi
2.25	F04_5_Ausgabe_Datei	xxxii
2.26	HF_02_0_Extrudieren	xxxiii
2.27	HF_06_0_Auffuellen.....	xxxiv
2.28	HF_12_Randvoxel_bestimmen	xxxv
	Eidesstattliche Erklärung.....	xxxvi
	Literaturverzeichnis	Fehler! Textmarke nicht definiert.

NOMENKLATUR

Fachbegriffe und Abkürzungen

Anstoßkante	gemeinsame Kante zweier benachbarter Shells
Automatisierungsgrad	Maschineller Fertigungsanteil
Balkentheorie	Vereinfachtes Berechnungsmodell für Verformungen
Bezierfunktion	Funktion zur parametrischen Kurvenmodellierung
CAD	Computer-Aided-Design
CFD	computational fluid dynamics
Drapierprozess	geordnete Ablage von textilen Geflechten
Einheitszelle	als homogen anzusehendes Element eines Faserverbundbauteils
Elastizitätsmodul	Materialkennwert der den Zusammenhang aus Dehnung und Spannung beschreibt
Extrusionskörper	Der im Rahmen eines Extrusionsprozesses entstandene Körper, hier: die Gesamtheit aller zu einer Shell gehörigen Voxel
FE	Finite-Elemente
Filament	Elementares Objekt einer Faser
Filamentgehalt	Volumenanteil an Filamenten in einer Faser
Finite-Elemente	Berechnungsverfahren mittels Diskretisierung
"Full-Data-Voxel"-Modell	Abbildung aller geometrischen und mechanischen Eigenschaften eines Voxel
Halbzeug	Vorbearbeitete Werkstücke
Harz	Material zur Fixierung und Verbindung der Fasern
ID	Identifikationsnummer
Kavität	Volumen, das durch das Harz gefüllt wird
Knoten	hier: Punkt mit eindeutiger Position im Raum
Kollisionskörper	Menge aller zusammenhängender kollidierender Voxel
Kontaktfläche	Fläche, an welcher sich mehrere Fasern berühren
Lastspitzen	Aufgrund der Berechnungsmethode auftretende hohe Belastungen
Lotfußpunkt	hier: Schnittpunkt des Lots an der Spiegelgeraden
Matlab	zur Entwicklung des Programmes genutzte DIE
maximale Dehnung	maximal mögliche Dehnung ohne dass Materialversagen eintritt
Meshing	Gittererzeugung
mesoskopisch	Größenordnung zwischen mikroskopisch und makroskopisch
Node	siehe Knoten
Permeabilität	Durchlässigkeit
Pfad	hier: Weg entlang der ein Querschnitt extrudiert wird
Positionsindex	Eindeutiger Index der sich aus den Positionskoordinaten ergibt
Preform	Faserverbundbauteil im RTM-Prozess vor der Harzinjektion
Querkontraktionszahl	Materialkonstante
Referenzursprung	hier: der Ursprung der Eingabedaten
Roving	Bündel aus Filamenten, auch als Faser bezeichnet
Rovingquerschnitt	Querschnitt des Rovings normal zur Faserrichtung
RTM-Prozess	Resin-Transfer-Moulding, Fertigungsprozess für Faserverbundbauteile
Shell	Durch Eckknoten definierte Fläche
Spline	siehe Pfad
Torsionsmodul	Materialkonstante die die Verformung bei Torsion beschreibt

Voxel	Volumen-Pixel
Voxelmodell	Menge aller Voxel
Werkzeug	hier: Das zur Harzinjektion genutzte Werkzeug
Zykluszeit	Fertigungsdauer

Formelzeichen

v	Richtungsvektor der Spiegelgeraden
v_x	Vektor in X-Richtung
$v_{x,1}$	Vorläufiger Vektor in X-Richtung
v_y	Vektor in Y-Richtung
v_z	Vektor in Z-Richtung
P_1	Erster Referenzpunkt
P_2	Zweiter Referenzpunkt
P_3	Dritter Referenzpunkt
V_f	Durchschnittlicher Faseranteil des Voxel
ν_f	Faseranteil
A_f	Durchschnittliche Faserrichtung des Voxel
a_f	Faserrichtung
V	Voxelvolumen
Θ	Garnrotationswinkel
h	Garnhöhe
s	Abstand zwischen den Garnpfaden benachbarter Garne
S	Garnpfad
U	Hilfsvektor in Richtung der Höhe
X'	Richtung des Garnquerschnitts in X
Y'	Richtung des Garnquerschnitts in Y

ABBILDUNGSVERZEICHNIS

Kommentiert [CD7]: Quellen entfernen

Abbildung 2.1-1: Data Flow [13] S. 1872	4
Abbildung 2.2-1: Möglichkeiten WiseTex [18]	5
Abbildung 2.2-2: Übersicht WiseTex Suite [18]	6
Abbildung 2.2-3: Garnaufbau WiseTex aus Pfad und Querschnitt [18]	6
Abbildung 2.2-4: Garnpfade [18]	7
Abbildung 2.2-5: Garnverlauf zwischen den Knoten [18]	7
Abbildung 2.2-6: Erzeugung gewobenes Geflecht [18]	8
Abbildung 2.2-7: Vergleich simulierter vs. Realer Garnverlauf [18]	8
Abbildung 2.2-8: Übersicht Geflechte [22] S. 32	9
Abbildung 2.2-9: Aufbau Texgen [20] S. 32.....	10
Abbildung 2.2-10: Querschnitt TexGen [20] S. 49.....	10
Abbildung 2.2-11: Koordinatensysteme TexGen Querschnitt [20] S. 19	11
Abbildung 2.2-12: interpolierte Querschnitte [20] S. 20.....	11
Abbildung 2.2-13: Polygone [20] S. 23	12
Abbildung 2.2-14: Querschnitt mesh [20] S. 24	12
Abbildung 2.2-15: Kontaktart [20] S. 154	13
Abbildung 2.2-16: Kreuzung [20] S. 155	13
Abbildung 2.2-17: Verformung durch Rotation [20] S. 55	14
Abbildung 2.3-1 Aufbau virtuelles Modell.....	15
Abbildung 2.3-2: Dimensionen der verschiedenen Modellierungsebenen 2008_COMPUTATION OF THE P [25].....	16
Abbildung 2.3-3: Workflow Fasersimulation Lomov et al [23] S. 3	16
Abbildung 2.4-1:Prozess Voxelerzeugung 2008_COMPUTATION OF THE P (2) [25]	16
Abbildung 2.5-1: doi 10.1016/j.3d.2008.02.039.....	17
Abbildung 2.5-2: Middleton - Geometric and (10) [20] S. 49	17
Abbildung 3.1-1 Ortsvektor zur Bestimmung der Voxelposition.....	18
Abbildung 3.1-2 Verschiebung des Gitterursprungs	19
Abbildung 3.1-3 Skizze Rovinghöhe	19
Abbildung 3.1-4 Skizze Kantenlänge Voxel	20
Abbildung 3.2-1 Visualisierte Eingabedaten.....	21
Abbildung 3.2-2 Visualisierte aufbereitete Eingabedaten.....	21
Abbildung 3.2-3 Extrudierter Roving in Draufsicht inkl. Shells	22
Abbildung 3.2-4 Darstellung Übergangsproblematik Seitenansicht	22
Abbildung 3.2-5 Darstellung Übergangsproblematik Querschnitt	22
Abbildung 3.2-6 Darstellung durch die Übergangsfunktion verschobener Voxel.....	23
Abbildung 3.2-7 Darstellung durch die Umformfunktion verschobener Voxel.....	23
Abbildung 3.2-8 Notation Knoten Input	23
Abbildung 3.2-9 Notation Shell Input	24
Abbildung 3.2-10 Darstellung eines einzelnen Rovings inkl. Knoten-IDs	24
Abbildung 3.2-11 Darstellung eines einzelnen Rovings inkl. Shell (weiß) und Knotenreihenfolge (schwarz).....	26
Abbildung 3.2-12 Darstellung Referenzgitter inkl. Faserdaten	27
Abbildung 3.2-13 Ableitung des Koordinatensystems aus der Shell	27
Abbildung 3.2-14 Darstellung der Matrix über der Shell-Grundfläche	28
Abbildung 3.2-15 Darstellung der extrudierten Halbellipse über der Shell-Grundfläche	29
Abbildung 3.2-16 Skizze Extrusionsverfahren.....	30
Abbildung 3.2-17 Visualisierung der extrudierten Halbellipse über der Shellfläche	30
Abbildung 3.2-18 Skizze der Draufsicht auf den Extrusionskörper	30
Abbildung 3.2-19 Skizze Beschneideverfahren des Extrusionskörpers.....	31
Abbildung 3.2-20 Darstellung des finalen Extrusionskörpers	31
Abbildung 3.2-21 Visualisierung einer Extrudierten Shell	32
Abbildung 3.2-22 Zustand vor der Spiegelung. Die kollidierenden Voxel sind in Gelb dargestellt.....	33
Abbildung 3.2-23 Bestimmung der Spiegelgeraden.....	34
Abbildung 3.2-24 Skizze Spiegelverfahren	34

Abbildung 3.2-25 Spiegelung an der Geraden	35
Abbildung 3.2-26 Zustand nach der Spiegelung. Die vormals kollidierenden Voxel sind in Gelb dargestellt.....	35
Abbildung 3.2-27 Gliederung und Zuordnung der kollidierenden Voxel.....	36
Abbildung 3.2-28 Vorgehensweise der Umformung.....	36
Abbildung 3.2-29 Faserverbund Querschnitte (11)	37
Abbildung 3.2-30 Faserverbund Querschnitte 2 (13)	37
Abbildung 3.2-31 Aufbau eines Solids	37
Abbildung 3.2-32 Ermittlung eines Kollisionskörpers	38
Abbildung 3.2-33 Skizze der Spiegelung an der Schnittebene	39
Abbildung 3.2-34 Skizze der Scheibenermittlung	40
Abbildung 3.2-35 Ermittlung der pot. freien Positionen	40
Abbildung 3.2-36 Ermittlung der freien Positionen.....	41
Abbildung 3.2-37 Horizontale Verschiebung.....	41
Abbildung 3.2-38 Skizze der neuen Randvoxel	42
Abbildung 3.2-39 Problematik bei anliegenden fremden Rovingen	42
Abbildung 3.2-40 Vertikale Verschiebung.....	43
Abbildung 3.3-1 Visualisierung der Eingabedaten	44
Abbildung 3.3-2 Visualisierung der in Dreiecke konvertierte Eingabedaten	44
Abbildung 3.3-3 Randflächen (blau) inkl. erzeugter Voxel	45
Abbildung 3.3-4 Querschnitt der Randflächen inkl. der erzeugten Voxel	45
Abbildung 3.3-5 Befüllungsvorgang inkl. Projektionsfläche.....	46
Abbildung 3.3-6 Darstellung möglicher "Löcher" in erzeugten Voxelhülle	46
Abbildung 3.3-7 Darstellung der In-/Outletflächen inkl. Harzinjektionsrichtung	47
Abbildung 3.4-1 Aufbau eines Solids	48
Abbildung 3.4-2 Darstellung der Randflächen vor der Filterung	49
Abbildung 3.5-1 Vergleich Arbeitsspeicherauslastung Matrix-Liste.....	54

TABELLENVERZEICHNIS

Tabelle 3.1-1 Liste der Voxel	18
Tabelle 3.1-2 Liste der Voxeltypen	19
Tabelle 3.1-3 Auswahl Injektionsrichtung	20
Tabelle 3.2-1 Ursprüngliche Übersicht des Knoten- und Shellstatus	25
Tabelle 3.2-2 Übersicht Knoten- und Shellstatus nach erfolgter Shellzuordnung	25
Tabelle 3.2-3 Voxelliste die durch die Extrusion erzeugt wurde	32
Tabelle 3.4-1 Solids Tabelle zweier aufeinanderliegender Voxel	48
Tabelle 3.4-2 Knotenliste in Einheit Kantenlänge	49
Tabelle 3.4-3 Knotenliste in Einheit mm	49
Tabelle 3.4-4 Randflächenliste	49

UNTERSUCHUNG

1 Einleitung

„*DER SCHWERKRAFT MIT LEICHTIGKEIT BEGEGNEN.*“

[1]

Dieser Slogan bewirbt den im Jahr 2015 vorgestellten 7er BMW. Als eines der ersten Großserienfahrzeuge verfügt der Wagen über tragende Elemente aus carbonfaserverstärktem Kunststoff. Diese werden als sogenannter „Carbon Core“ vermarktet und sollen sowohl einen geringen Verbrauch garantieren als auch ein sportlich-dynamisches Fahrgefühl vermitteln [2] S. 2. Bei vergangenen Fahrzeuggenerationen standen diese beiden Eigenschaften stets im Widerspruch zueinander. Kurze Beschleunigungszeiten und hohe Endgeschwindigkeiten wurden bisher durch hohe Motorleistungen bei ebenfalls hohem Kraftstoffverbrauch erreicht [3] S. 13. Dieser Weg ist jedoch aufgrund gesetzlicher Vorgaben und geänderten Kundenanforderungen aktuell nicht mehr gangbar [4] S. 193. Während Regulierungsbehörden weltweit Abgas- und Verbrauchsvorschriften aus Umweltschutzgründen verschärfen [5] S. 16, verlangen die Kunden, aufgrund gestiegener Kraftstoffpreise und einem veränderten Umweltbewusstseins sowohl nach verbrauchsarmen, als auch weiterhin sportlichen, sicheren und luxuriösen Fahrzeugen. Neben der Entwicklung und Herstellung von verbesserten Motoren versucht die Automobilindustrie diese vermeintlichen Gegensätze durch eine Reduktion der Fahrzeuggewichte aufzulösen [3] S. 38. In einem ersten Schritt wurden deshalb Karosseriebauteile aus Aluminium und hochfesten, leichten Stählen gefertigt. Der nächste Evolutionsschritt stellt nun die Verwendung von Bauteilen aus Carbonfaserverbund im PKW-Rohbau dar [5] S. 20. Während der Einsatz von Carbonfasergeweben in anderen Industrien wie der Luft- und Raumfahrt, bei High-End-Sportgeräten und im Automobilrennsport aufgrund der überlegenen mechanischen Eigenschaften bereits seit längerer Zeit beliebt und erprobt ist, fand der Werkstoff in der Massenfertigung durch die hohen Entwicklungs- und Fertigungskosten bisher keine Anwendung [5] S. 21–22. Diese im Vergleich zu Bauteilen aus Metall oder Kunststoff erhöhten Kosten entstehen hauptsächlich aufgrund geringer Automatisierungsgrade, einem hohen Aufwand für die Qualitäts sicherung, da Werkstoffdefekte nur schwerlich zu erkennen sind, und dem Bedarf einer lastpfadkorrekten Entwicklung der Bauteile in Verbindung mit weniger vorhandenen Erfahrungswerten in der Konstruktion und Auslegung [6] S. 17–18. Für den Durchbruch in preissensitiven Branchen, wie der Automobilbranche, fehlt es deshalb an prozesssicheren und automatisierten Entwicklungs- und Fertigungsverfahren [7] S. 9. Die Fertigung von CFK-Bauteilen erfolgt derzeit meist im sogenannten RTM-Verfahren (Resin-Transfer-Moulding). Das Verfahren zeichnet sich durch kurze Zykluszeiten und die Herstellbarkeit großer, komplexer Bauteile aus [8] S. 35. Für das Verfahren werden Faserhalbzeuge entkernähnlich in einer Form drapiert. Daraufhin wird Harz unter Druck in das Gelege injiziert. Abschließend wird das Bauteil durch eine Presse zur Gewichtsreduktion verdichtet, indem überschüssiges Harz aus dem Bauteil herausgepresst wird [8] S. 35–36. Die Platzierung der Faserhalbzeuge geschieht entweder klassisch durch das manuelle oder teilautomatisierte Verlegen von Fasermatten auf einer Form, oder dem umweben eines Körpers [9] S. 30–32. Verglichen mit der Verwendung von Fasermatten, sogenannten Axial- oder Multiaxialgelegen, verfügt das Webverfahren oftmals über einen höheren Automatisierungsgrad, einer damit einhergehenden besseren Prozesssicherheit und niedrigen Werkstoffkosten durch äußerst geringen Verschnitt [9] S. 318. Nachteilig sind hingegen die mechanischen Eigenschaften der Flechtkörper aufgrund der Ondulation der einzelnen Fasern im Gewebe und die im Vergleich zu Drapierverfahren geringe Flexibilität in Bezug auf herstellbare Formen. Aus diesen Gründen wird das Flechtnverfahren hauptsächlich für Hohlkörper, wie beispielsweise Wellen eingesetzt [9] S. 318, während die Legeverfahren für Flächenbauteile, wie Motorhauben oder Windradflügel, verwendet werden. Die mechanischen Eigenschaften von Faserverbundbauteilen hängen jedoch, neben der belastungsgerechten Platzierung der Fasern, auch von der ordnungsgemäßen Durchdringung des Bauteils mit der jeweiligen Matrix ab [10] S. 73. Dies resultiert aus der geringen Fehltoleranz von Faserverbundbauteilen, die zu Delaminierung und darauffolgendem Bauteilversagen führen kann [6] S. 17. Aus diesem Grund muss die vollständige Durchdringung des Geflechts bereits im Entwicklungsprozess sichergestellt werden. Entscheidend ist hierfür die sogenannte Permeabilität, die die Durchlässigkeit des Fasergeflechts für das injizierte Harz

beschreibt [11] S. 260. Um den Kosten- und Zeitaufwand für die Entwicklung und Konstruktion von Faserverbundbauteilen zu minimieren, wird aktuell versucht auf reale physische Prototypen und Testreihen zu verzichten und diese durch Computersimulationen zu ersetzen [9] S. 564. Die Realisierung eines solchen vollkommen digitalen Modells ist eines der Forschungsprojekte unter dem Dach der ARENA2036. Bei der ARENA2036 handelt es sich um einen Forschungscampus an der Universität Stuttgart, an dem verschiedene Partner aus Wirtschaft und Wissenschaft, unter anderem das Institut für Flugzeugbau, Leichtbaukonzepte für kommende Fahrzeuggenerationen entwickeln. Das Ziel dieser Bachelorarbeit ist somit vorhandene Lücken im Prozess zu einem vollkommenen digitalen Prototyp für Faserverbundbauteile zu schließen, um diese Bauteile zukünftig bei geringen Kosten und einer gleichbleibenden hohen Qualität in sicheren, umweltfreundlichen und komfortablen Großserienfahrzeugen einzusetzen [12] S. 506.

Mein persönliches Interesse an diesem Thema röhrt aus meiner Faszination für Autos und Leichtbau, die ich während meiner Jugend und einem Praktikum bei der Daimler AG entwickelte. Im Rahmen dieser Arbeit konnte ich diese Faszination, mit den im Studium erlernten Inhalten, aus den Fachgebieten der Informatik, der Werkstoffkunde, dem Leichtbau und dem technischen Projektmanagement verbinden und bestehende Fähigkeiten sowohl vertiefen als auch praktisch anwenden.

Begonnen wurde die Arbeit mit einer Literaturrecherche, die das Ziel hatte, vorhandene Lücken im Prozess einer vollkommenen digitalen Modellierung von Faserverbundbauteilen zu identifizieren. In diesem Rahmen wurden deshalb zunächst bereits vorhandene Softwareprodukte und Lösungsansätze analysiert. Diese wurden auf bestehende Unzulänglichkeiten hin untersucht. Speziell betrachtet wurden die beiden Softwares Wisetex und Texgen der Universitäten Leuven und Nottingham.

Die Ergebnisse der Literaturrecherche werden in Kapitel 2 „Stand der Technik“ erläutert. Vorgestellt werden die Softwareautoren, der aktuelle Entwicklungsstand und die Entwicklungshistorie, Einschränkungen in Bezug auf das angepeilte Nutzungsszenario in der Automobilfertigung und Entwicklung, bereits in Diskussion befindliche Weiterentwicklungen, der allgemeine Aufbau der Softwaren, im speziellen das Verhalten bei Faserkollisionen und die Anforderungen an, bzw. der Aufbau der Ein- und Ausgabedateien. Darauffolgend werden für die gefundenen Probleme bestehende Lösungskonzepte und bereits definierte Anforderungen aufgezeigt. Diese werden gemäß ihren Dimensionen in die mikro-, meso- oder makroskopische Ebene eingeteilt. Außerdem wird das Konzept der Voxeldiskretisierung vorgestellt. Neben der Definition der Voxel werden im Rahmen des Kapitels 2 auch bereits umgesetzte Softwareanwendungsfälle von Voxel präsentiert und deren Vor-, sowie Nachteile gegenüber anderen Lösungen herausgearbeitet. Abgeschlossen wird das Kapitel mit der Vorstellung von Forschungsarbeiten, die sich mit der Verformung von Fasern innerhalb eines Geflechts unter Druck- und Spannungseinflüssen beschäftigen.

Das im Kapitel 3 präsentierte Modell basiert auf den Ergebnissen der Literaturrecherche. Das Kapitel enthält die Beschreibungen der programmierten Funktionen. Begonnen wird mit einer Vorstellung des Konzepts des geordneten Gitters das den betrachteten Raum beschreibt und die Implementierung des Voxelmodells. Des Weiteren beschreibt der erste Abschnitt des Kapitels 3 „Untersuchung“ die zum Austausch zwischen den Unterfunktionen genutzten Schnittstellen und Datenstrukturen. Auch die vom Nutzer geforderten Eingaben und die angewandten Prinzipien der Programmierung werden in diesem Abschnitt beschrieben. Der zweite Abschnitt der Untersuchung beschreibt die Verarbeitung der Faserdaten. Zunächst wird die Aufbereitung der eingelesenen Daten und die Modellierung des Referenzraumes erläutert. Anschließend wird die Extrusion der elementaren Körper dargestellt. Nachfolgend wird die Erstellung der Übergänge und die verwendete Lösung zur Kollisionsbeseitigung vorgestellt. Die Verarbeitung der Kavitätsdaten wird im dritten Abschnitt des Kapitels erläutert. Wie bereits im vorherigen Abschnitt wird zunächst die Aufbereitung der eingelesenen Daten geschildert. Daraufhin wird die Konvertierung der Randflächen in Voxel und die Befüllung des beschriebenen Raumes mit Voxeln vorgestellt. Abgeschlossen wird der Abschnitt mit der Ermittlung der Ein- und Auslassflächen für die Harzinjektion. Die Präsentation der implementierten Ausgabefunktionen bildet den vierten Abschnitt. Das Ende des Untersuchungskapitels 3 bildet die Darstellung der genutzten Optimierungsfunktionen und eine detaillierte Laufzeitbetrachtung. Der allgemeine Funktionsaufbau wird zu Beginn jeder beschriebenen Funktion erklärt, bevor die Beschreibung der tatsächlichen Umsetzung, unterstützt durch zahlreiche Abbildungen, erfolgt. Diese Beschreibung orientiert sich nah am tatsächlichen Programmcode, bildet jedoch einige Details aus Gründen der Übersichtlichkeit nicht

ab. Eine detaillierte Beschreibung des Programmablaufs anhand von Flussdiagrammen befindet sich im Anhang.

2 Stand der Technik

2.1 EINORDNUNG IN DEN WORKFLOW

Der Modellierungsprozess von Faserverbundbauteilen erfolgt typischerweise entlang des Mikro-Meso-Makro-Ebenenmodells. Aus dem Mikromodell werden die Materialeigenschaften der Garne für das Mesomodell abgeleitet, das durch die Abbildung der Einheitszelle wiederum als Basis für das Makromodell dient [13] S. 1870. Gemäß diesem Modell simuliert die Software PAM-Crash den Flechtprozess und erzeugt das makroskopische Modell. Auf Basis dieses Modells sollen anschließend die mechanischen Eigenschaften des Bauteils mittels FE-Simulation ermittelt werden [14] S. 2. Diese Simulation basiert gemäß Abbildung 2.1-1 auf den Daten der Meso-Ebene, die wiederum aus der Garn/Rovinggeometrie, und den als einheitlich angenommenen Eigenschaften der Mikroebene erzeugt wird.

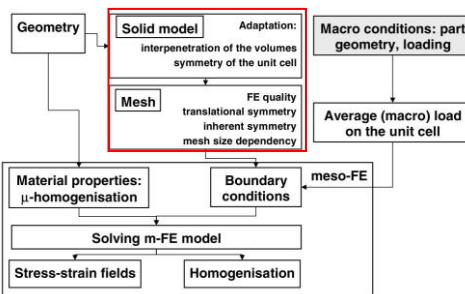


Abbildung 2.1-1: Data Flow [13] S. 1872

Die Mesoebene bildet gemäß Abschnitt 2.3 die Einheitszellen ab. Die Erzeugung der Geometrie der Einheitszellen bei den Dreidimensionalen Flechtkörpern, die in dieser Arbeit betrachtet werden, geschieht momentan in zwei Schritten. Im ersten Schritt wird der Faserquerschnitt, unabhängig von der Realität, so gewählt, dass es keine Kollisionen zwischen den Garnen gibt. Entlang der so ermittelten Garnpfade wird im zweiten Schritt der tatsächliche Querschnitt extrudiert. Überschneidungen zwischen den Garnen werden hierbei akzeptiert [14] S. 28. Um auch komplexe Einheitszellenarchitekturen ohne das aufwendige Erstellen von Meshes verarbeiten zu können, werden die Fasern durch Voxel diskretisiert [15] S. 60. Neben der Ermittlung der mechanischen Eigenschaften dient das Modell auch zur Injektionssimulation. Hierfür werden die Voxel des betrachteten Raumes entweder dem Faservolumen oder dem Matrixvolumen zugeordnet. Der Aufbau innerhalb der Fasern, auf Mikroebene, wird sowohl zur Eigenschafts-, als auch zur Permeabilitätsanalyse als homogen angenommen und dementsprechend separat eingegeben [15] S. 59.

2.2 VORHANDENE SOFTWARE

Derzeit werden für die mesoskopische Simulation der Preform-Fasern im RTM-Prozess entweder Finite-Elemente-Löser (FE-Solver) wie ANSYS, LS-DYNA, ABAQUS oder NX Nastran in Kombination mit einem CAD-Programm (z.B. CATIA) oder spezielle Programme zur Simulation von textilen Geflechten genutzt. Die Letztgenannten bieten den Vorteil der integrierten Geometrieerzeugung. Damit einher geht die Möglichkeit, die Attribute des Gewebes einfach anzupassen. Aktuell werden die Softwarelösungen WiseTex Suite und TexGen vorrangig verwendet und daher in den folgenden Abschnitten vorgestellt [16].

2.2.1 Wisetex

Die Software Wisetex wurde am Institut für Materialwissenschaften der KU Leuven von Prof. Stepan Lomov, Ir. Sergej Kondratiev, Ir Maarten Mosen, Ir. Teo Peeters, Ir. Andreas Prodromou und Prof. Ignnaas Verpoest entwickelt [17] S. 7. Wisetex basiert teilweise auf der Software CETKA (Russisch für Netz). CETKA entstand ab 1991 an der Universität für Technologie und Design Sankt Petersburg. In Zusammenarbeit mit der KU Leuven kann die Software seit 1999 auch mikromechanische Eigenschaften der verwendeten Fasern in das Modell einbinden und verarbeiten. [17] S. 8. CETKA simuliert den inneren Aufbau von mehrschichtigen Faserverbundgeflechten und kann somit die geometrische Struktur der Fasern wiedergeben. Dafür werden auch die Kontakte zwischen den einzelnen Fasern mittels des Prinzips der minimalen Energie bezüglich ihrer mechanischen und geometrischen Attribute abgebildet. [18] S. 1. Die Entwicklung von WiseTex begann 1999. Zunächst konnten zwei- und dreidimensionale Modelle von Geflechten durch das Programm erzeugt werden. Der Funktionsumfang wurde mit der Zeit auf den im Folgenden beschriebenen erweitert, indem zunächst die Verarbeitung von weiteren Geflechtypen, und ab 2002 Verformungen in das Programm implementiert wurden. [17] S. 8. Als Eingabedaten verwendet Wisetex die mechanischen Eigenschaften der verwendeten Fasern (Kompressions-, Biege- und Spannungsverhalten), den Flechtaufbau, und die Form der Kontaktflächen zwischen den Garnen [19] S. 2.

Wisetex kann verschiedene Geflechte hinsichtlich Geometrie und mechanischer Eigenschaften modellieren. Die Software berücksichtigt hierfür die physikalischen Eigenschaften der Garne [20] S. 1690. Das Programm kann zur Modellierung von Fasern, Garnen, (gestrickten(d)) Geweben(f), Stoffen(b), Flechtgeweben(c), Unidirektionalen Preform-Halbzeugen(e) und Lamine(a) verwendet werden [17] S. 7. Die verschiedenen Textilarten sind in Abbildung 2.2-1 ersichtlich. Alle Modelle basieren auf der Simulation einer Einheitszelle [19] S. 1.

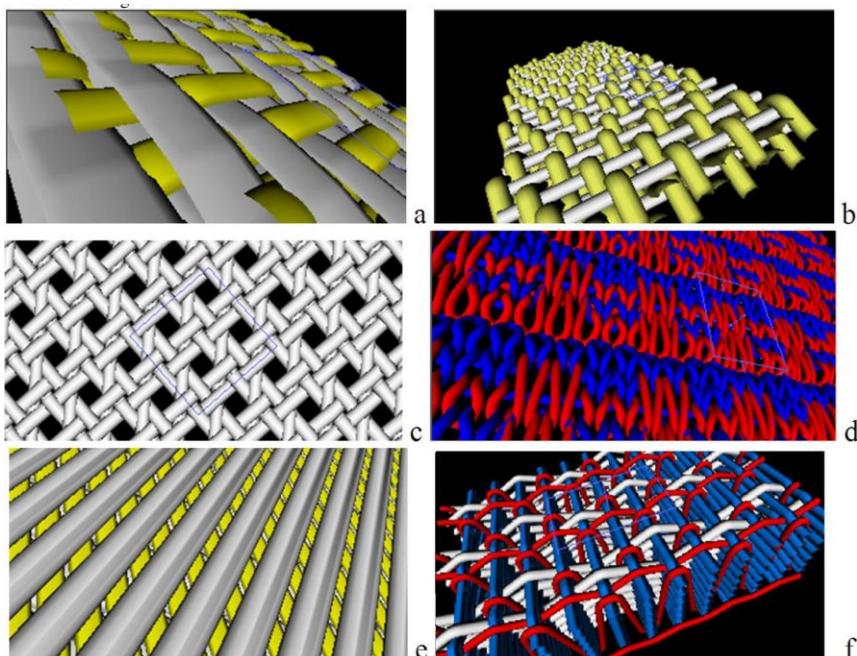


Abbildung 2.2-1: Möglichkeiten WiseTex [19] S. 3

Um die Software Wisetex entstand die WiseTex Suite, die neben Wisetex selbst aus weiteren Anwendungen zur Modellierung, Darstellung und Simulation von Textilien besteht. Eine Übersicht über die weiteren Anwendungen gibt Abbildung 2.2-2.

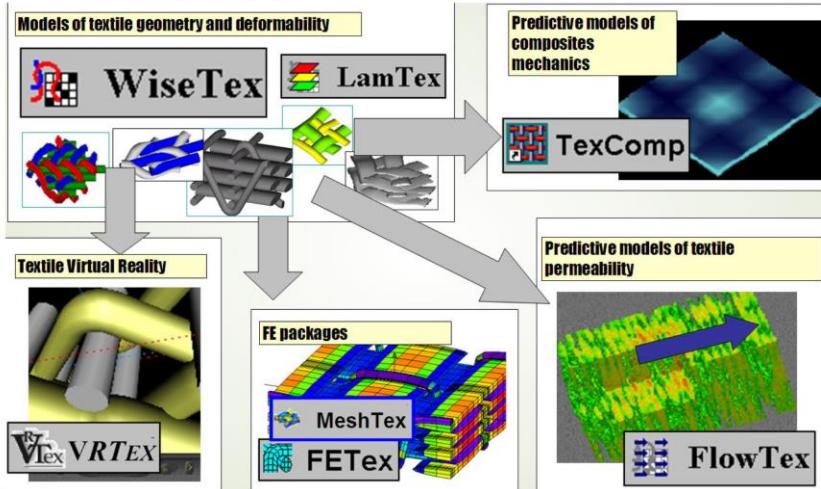


Abbildung 2.2-2: Übersicht WiseTex Suite [19] S. 19

Das Modul LamTex wird zur Modellierung von Laminaten eingesetzt. Die beiden Module MeshTex bzw. FETex exportieren die erzeugten Daten in Dateien zur FE-Simulation in den Programmen ANSYS bzw. SACOM FE. Durch TexComp können Steifigkeiten analytisch ermittelt werden. FlowTex ermöglicht Injektionssimulationen auf Basis von Permeabilitätsanalysen und VRTex wird zur Visualisierung der Modelle in Virtual-Reality-Umgebungen eingesetzt [21] S. 9.

Wie in Abbildung 2.2-3 veranschaulicht, werden die Garne in WiseTex durch die Extrusion eines Querschnitts entlang eines vorgegebenen Pfades erzeugt.

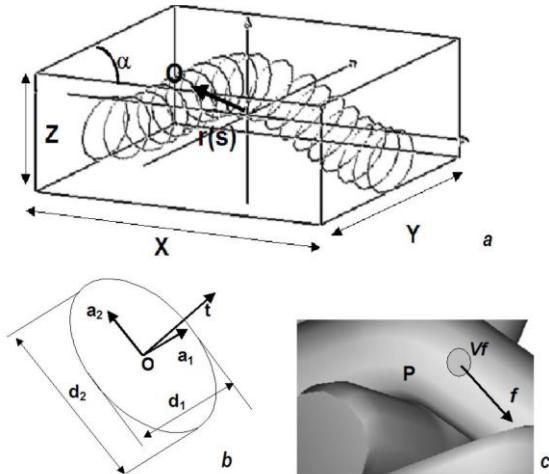


Abbildung 2.2-3: Garnaufbau WiseTex aus Pfad und Querschnitt [19] S. 2

Die Garnpfade werden bestimmt, indem die zu betrachtende Einheitszelle in Kreuzungspunkte eingeteilt wird. Diese Kreuzungspunkte beschreiben innerhalb eines Gitters mit den Ausmaßen der Einheitszelle den Verlauf eines Garnpfades. Ein Kreuzungspunkt ergibt sich durch den Kontakt von mehreren Garnen. Da die Einheitszellen regelmäßig aufgebaut sind, können durch dieses Verfahren die Garnpfade relativ zueinander beschrieben werden [19] S. 3. Dies wird in Abbildung 2.2-4 anhand einer Faserebene

dargestellt. Aus der Abbildung links ergibt sich die rechte Matrix, die für jedes Garn die diskretisierte Position in Z-Richtung wiedergibt. Die Oberfläche des Textils entspricht Ebene 0.

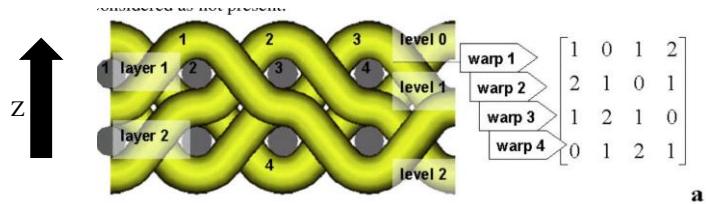


Abbildung 2.2-4: Garnpfade [19] S. 4

Die Pfadgeometrien zwischen den Knotenbereichen werden durch das minimale Energie-Prinzip festgelegt. Entscheidend ist hierfür der Abstand der Knoten in X-, respektive Y-Richtung und der in Z-Richtung innerhalb dieses Abstands zurückzulegende Weg. Hieraus ergeben sich die auf das Garn einwirkende Biegekräfte (siehe Abbildung 2.2-5) [19] S. 5.

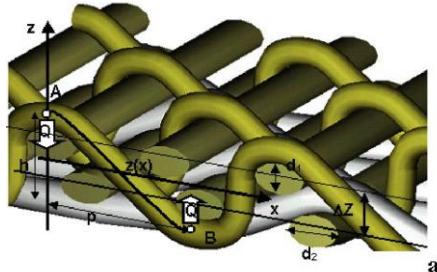


Abbildung 2.2-5: Garnverlauf zwischen den Knoten [19] S. 5

An den Kontaktstellen üben die Garne gegenseitig Kräfte aufeinander aus. Deshalb kommt es zu Verformungen des Garnquerschnitts und zu einer Kompaktierung in Z-Richtung des gesamten Geflechts. Die hierbei übertragenen Verformungsenergien tragen gemäß dem Prinzip der minimalen Energie ebenfalls zur Verformung der Garne an und zwischen den Kontaktstellen bei [19] S. 6. Der Querschnitt der Garne wird durch zwei Variablen bestimmt, die die Ausmaße in zwei festgelegte Richtungen und der Verdrehung entlang des Querschnitts angeben. Diese Verdrehung entsteht beispielsweise durch Kontakte mit anderen Garnen. Die Richtungen sind hierbei mit dem Garnpfad dauerhaft fest verbunden, sie drehen sich demzufolge entsprechend der angegebenen Verdrehung mit. Die Querschnittsform kann vom Nutzer flexibel vorgegeben werden, ist dann jedoch über den gesamten Pfad über konstant. Zusammenfassend werden die Garngeometrien durch die folgenden vier Variablen beschrieben: Koordinaten des Mittelpunktes des Querschnitts auf dem Garnpfad, die Richtung des Garnpfades, die Referenzrichtungen des Querschnitts und die Ausmaße des Querschnitts entlang dieser Referenzrichtungen [19] S. 2. Für geflochene Gewebe werden zunächst dieselben Algorithmen angewandt wie für gewobene Gewebe. Das Gewebe wird dementsprechend wiederum als regelmäßig angenommen. In einem weiteren Schritt wird das Gewebe durch die Kräfte an den Rändern umgeformt, wie in Abbildung 2.2-6 erkennbar ist. Diese zusätzliche Krafteinwirkung führt zu nochmals veränderten Garnpfaden. [19] S. 9

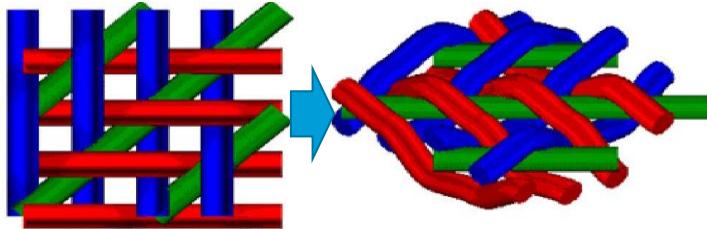


Abbildung 2.2-6: Erzeugung gewobenes Geflecht [19] S. 9

In Abbildung 2.2-7 werden die durch Wisetex erzeugten Garnverläufe mit den Realen durch übereinanderlegen verglichen. Erkennbar ist, dass die Garnverläufe nur teilweise korrekt ermittelt werden.

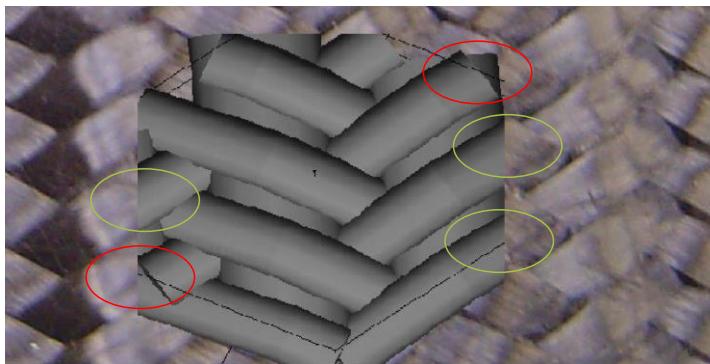


Abbildung 2.2-7: Vergleich simulierter vs. Realer Garnverlauf [19] S. 12

Wisetex erzeugt die Garngeometrien auf Basis von empirischen Untersuchungen. Diese Untersuchungen wurden mittels Elektronen-Mikroskopen an trockenen Geflechten durchgeführt [19] S. 1. Durch die auf die Garne wirkenden Kompressionskräfte werden zum einen die Garnverläufe zwischen den Knotenpunkten und zum anderen die Ausmaße der Querschnitte verändert. Des Weiteren ändern sich die als homogen betrachteten Eigenschaften der Fasern, da beispielsweise die Permeabilität durch die Verdichtung abnimmt. Für die Abbildung der Zugkräfte wird zunächst ein belastungsfreies Modell erstellt. Dieses wird je nach Wirkrichtung und Betrag der anliegenden Kräfte verformt. Die durch die Verzerrung des Modells auf die Garne neu einwirkenden Kräfte führen wiederum zu einer Kompaktierung des Geflechts und zu einer Veränderung der Garnpfade. Aus diesem Grund wird der Verzerrungsalgorithmus vor dem Kompressionsalgorithmus und der Garnpfadbestimmung durchgeführt. Die Simulation der Scherkräfte auf das digitale Modell geschieht ähnlich der Verzerrungssimulation. Ergänzt wird das Modell lediglich um die Reibung zwischen den Garnen. Die Reibkräfte ergeben sich aus den experimentell bestimmten Reibbeiwerten und den bereits bekannten Kräften zwischen den Garben [19] S. 16. Bei biaxialen Geflechten und Flechtwinkel ungleich 45° schneiden sich die Garne in der gemeinsamen Ebene aufgrund der vorgebenden Form der Kontaktflächen. Eine dynamische Anpassung der Formen ist jedoch nicht möglich. Aus diesem Grund wird entlang der Oberfläche der Garne ein sogenannter Hilfszyylinder modelliert. Die Form ergibt sich aus den noch unverformten Querschnitten entlang des Garnpfades [19] S. 10.

Wisetex kann in der aktuellen Version lediglich ebene Geflechte mit gleicher Faserrichtung verarbeiten. Komplexe Geflechtarchitekturen mit beispielsweise unterschiedlicher Garnanzahlen oder Richtungen in verschiedenen Ebenen können hingegen nicht modelliert werden [22] S. 4. So können beispielsweise, wie in Abbildung 2.2-8 ersichtlich wird, keine dreidimensionalen Flechtkörper erzeugt werden.

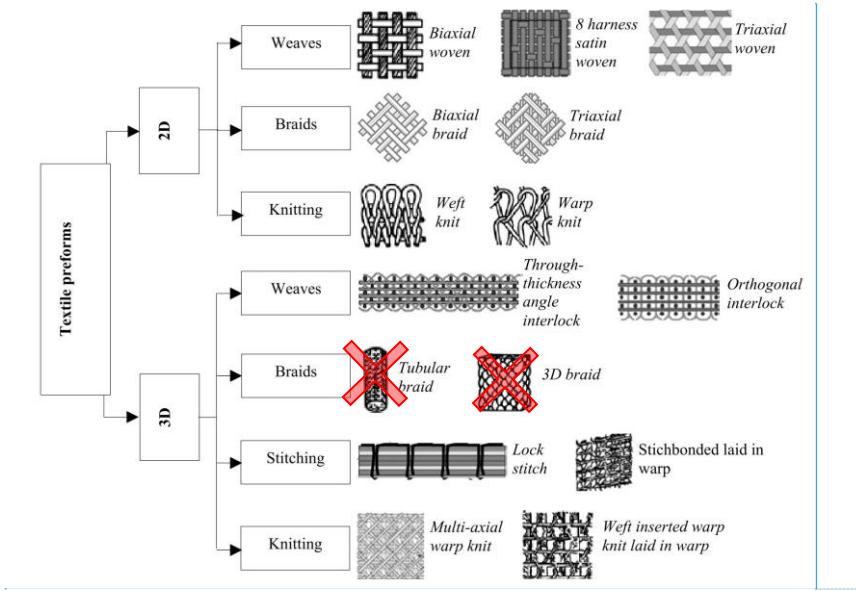


Abbildung 2.2-8: Übersicht Geflechte [23] S. 32

Des Weiteren kann der Kontaktalgorithmus von Wisetex nur Modelle korrekt abbilden, bei denen sich maximal 10% des gesamten Volumens aller Garne überschneiden [24] S. 13. Außerdem nimmt das Programm die Querschnittsflächen der Garne als symmetrische Körper an. Diese Annahme führt zu ungenauen Abbildungen durch die FE-Methode bezüglich des Garnvolumens. [20] S. 1690. Zukünftig soll Wisetex auch teilweise unregelmäßige Geflechte, in welchen beispielweise einzelne Garne fehlen, abbilden können. Hierzu muss der Code zur Bestimmung der Kreuzungen angepasst werden, da diese im Bereich der fehlenden Garne nicht existieren und somit kein Garnverlauf wie in Abbildung 2.2-4 ermittelt werden kann. [19] S. 4

2.2.2 TexGen

Souter begann 1998 mit der Entwicklung der ersten Version von TexGen. Aufgabe des Programmes war die Permeabilitätsanalyse und die Vorhersage von mechanischen Eigenschaften von textilen Werkstoffen. [21] S. 8. Die aktuelle Version von TexGen stammt von Martin Sherburn [21] S. 5. Sherburn begann mit der Arbeit an der zweiten Version im Jahr 2003. Aus Gründen der Wartbarkeit des Codes wurde dieser in der dritten Version von ihm abermals neu programmiert. [21] S. 8–9. TexGen ist in C++ geschrieben und unterstützt sowohl Linux als auch Windowsrechner [21] S. 31. Die Software simuliert die geometrischen und mechanischen Eigenschaften von textilen zwei- und dreidimensionalen Geflechten. Hierdurch dient die Software als Präprozessor für die FE-Simulation von Geflechten. Das Programm kann im Gegensatz zu WiseTex sowohl konstante als auch variable Garnquerschnitte verarbeiten. [20] S. 1689

Die Architektur besteht aus den fünf Modulen Core, Renderer, Export, Python-interface und GUI. Die Funktionen zur Erzeugung der Textilgeometrien befinden sich im Core Modul. Das Rendermodul erzeugt aus den Polygonen die grafische Darstellung, während das GUI-Modul die Benutzereingaben in einer grafischen Oberfläche aufnimmt. Das Exportmodul konvertiert und exportiert die erzeugten Datensätze zur weiteren Verarbeitung in kompatiblen Programmen. Das Python-Interface verbindet das Core-, Export- und das Rendermodul (siehe Abbildung 2.2-9). [21] S. 31. Das Coremodul behandelt die Garne, die zu untersuchende Einheitszelle und die Webstruktur als einzelne Klassen [21] S. 33.

Kommentiert [CD36]: Wie wird das Bild nach der Änderung zitiert?

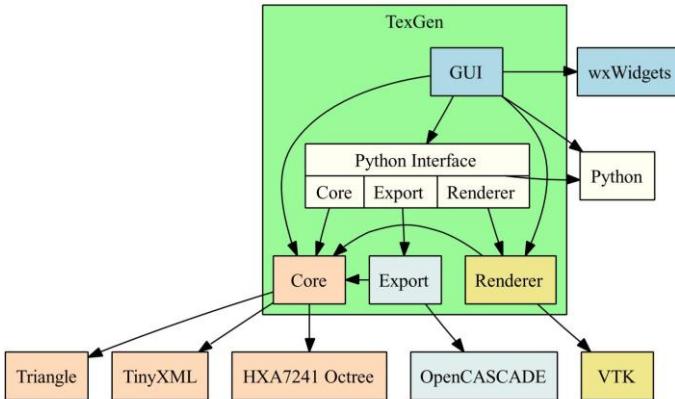


Abbildung 2.2-9: Aufbau TexGen [21] S. 32

Da TexGen lediglich Einheitszellen modelliert, werden die zu simulierenden Textilien zunächst in eben solche aufgeteilt. Die Größe dieser Zellen ist hierbei nicht vorgegeben. [21] S. 5. TexGen bestimmt die Garnpfade ähnlich wie WiseTex, indem der Garnverlauf in einzelne Splines zerlegt wird. Die Form der Splines wird durch die festgelegte Position von einigen Knoten bestimmt und durch eine Bezierfunktion beschrieben [21] S. 47. Entlang des Pfades wird wiederum ein festgelegter Querschnitt extrudiert [21] S. 11. Die einzelnen Fasern werden nicht modelliert, da die Garne aufgrund des andernfalls benötigten Rechenaufwandes als homogene Körper angenommen werden. [21] S. 5. Der Garnquerschnitt ist zweidimensional und wird entsprechend von zwei Variablen definiert. [21] S. 16. Die Form wird, wie in Abbildung 2.2-10 ersichtlich ist, als linsenförmig angenommen [21] S. 46.

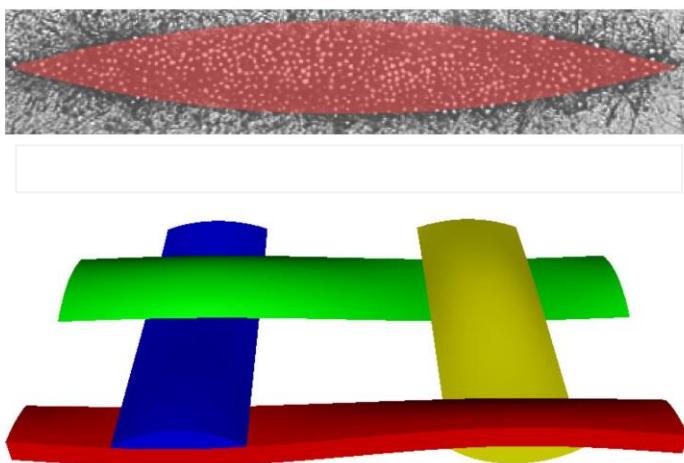


Abbildung 2.2-10: Querschnitt TexGen [21] S. 49

Die Querschnittsränder werden durch die Position ihres Mittelpunktes entlang des Pfades (S) und der festgelegten Querschnittsform (C) bestimmt. Für die absolute Orientierung wird ein Hilfsvektor eingeführt, der stets in Richtung der positiven Z-Achse zeigt (vgl. Abbildung 2.2-11). Dies stellt in den verarbeiteten Modellen die Höhe des Textilgeflechts dar. Der Vektor ist dementsprechend konstant. Die Orientierung der Querschnittsränder X' und Y' ergibt sich aus den folgenden Formeln [21] S. 18. Die Anordnung aller für den Garnquerschnitt relevanten Vektoren ist in Abbildung 2.2-11 dargestellt.

$$\vec{X}'(u) = \frac{\mathbf{S}'(u) \times \vec{U}}{\|\mathbf{S}'(u) \times \vec{U}\|}$$

$$\vec{Y}'(u) = \frac{\vec{X}'(u) \times \mathbf{S}'(u)}{\|\vec{X}'(u) \times \mathbf{S}'(u)\|}$$

Formel 2-1: Formeln Querschnitt [21] S. 18

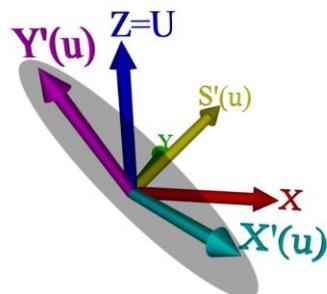


Abbildung 2.2-11: Koordinatensysteme TexGen Querschnitt [21] S. 19

Der Querschnitt der Garne kann für eine verringerte Rechnerauslastung entweder über den gesamten Pfad oder an einigen bestimmten Punkten entlang des Pfades festgelegt werden. Die Form der Querschnitte zwischen den Punkten entsteht durch lineare Interpolation. [21] S. 19 . Abbildung 2.2-12 stellt einen solchen interpolierten Garn dar. Die definierten Querschnittsflächen sind in der Abbildung rot markiert.

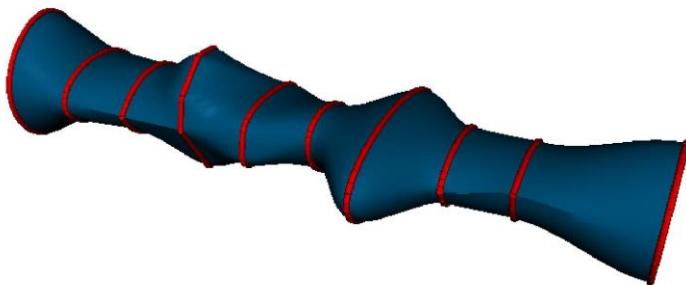


Abbildung 2.2-12: interpolierte Querschnitte [21] S. 20

Die Garnoberfläche wird aus Vierecken erzeugt, die wiederum aus jeweils vier Knoten definiert werden. Die Position der Knoten wird durch deren Lage auf dem Garnpfad (Blau) und dem Querschnittsrund (Orange) bestimmt (Abbildung 2.2-13). Die Anzahl der Knoten kann je nach Anforderung und zur Verfügung stehender Rechenleistung variiert werden. [21] S. 22–23.

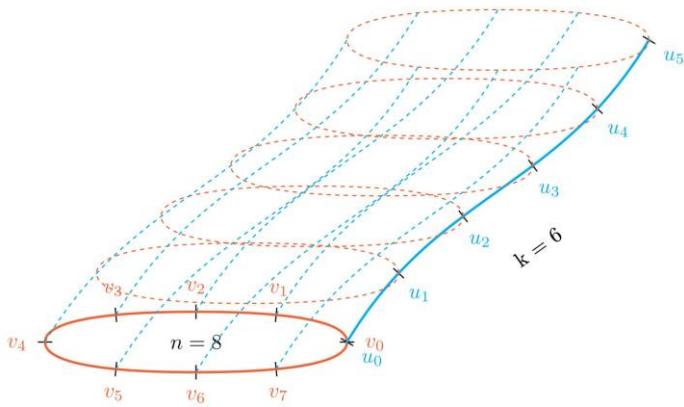


Abbildung 2.2-13: Polygonen [21] S. 23

Die Volumenkörper werden erzeugt indem zunächst die Querschnitte in ein zweidimensionales Mesh verwandelt und anschließend die benachbarten Querschnitte untereinander verbunden werden. Die Auflösung des benötigten Meshes wird vom Benutzer vorgegeben. Die Anzahl der Zellen muss jedoch aus Kompatibilitätsgründen auch bei variablen Querschnittsflächen über das gesamte Garn gleichbleiben. [21] S. 24.

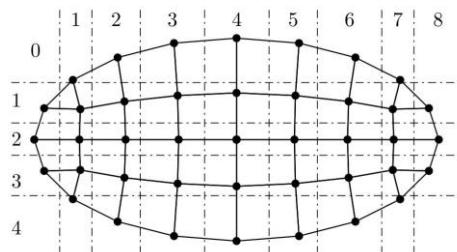
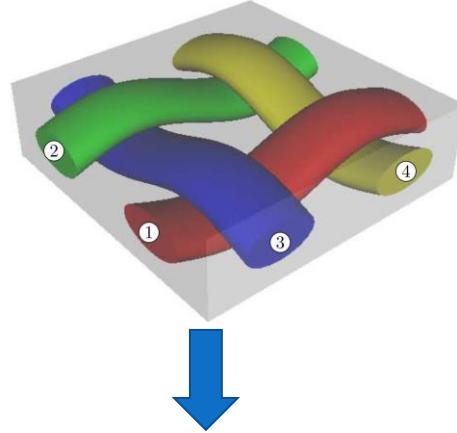


Abbildung 2.2-14: Querschnitt mesh [21] S. 24

Der Faseranteil wird innerhalb eines Querschnittsbereichs als konstant, und die Fasern selbst als inkompressibel angenommen. Der Faseranteil für einen Querschnitt ergibt sich aus der Summe der einzelnen Faserquerschnitte geteilt durch die Fläche des Garnquerschnitts. Die Anzahl der Fasern bleibt entlang des Garnes ebenfalls konstant. [21] S. 27. Die Form des Querschnitts ändert sich entlang des Garnpfades, da die Garne aufeinander gegenseitig Druck ausüben und sich deshalb verformen. [21] S. 18. Um den Bedarf an benötigter Rechenleistung und Speicherkapazität zu verringern, versucht TexGen für das gesamte Modell einen kleinsten gemeinsamen Garnabschnitt zu ermitteln. Das Modell der Einheitszelle wird danach aus diesem Referenzelement zusammengesetzt. [21] S. 21.

TexGen verfügt jeweils über einen Algorithmus zur Bearbeitung von parallel und normal zueinander verlaufenden Garnen. Dementsprechend werden die Kontaktstellen zunächst in einer Tabelle paarweise klassifiziert. [21] S. 153–154. Diese Klassifizierung ist anhand eines einfachen Geflechts in Abbildung 2.2-15 nachzuvollziehen. Die Garne können gemäß der Tabelle in der Abbildung entweder neben-, über- oder untereinanderliegen. Für die Lage relativ zu sich selbst wird N/A vermerkt.



	Yarn 1	Yarn 2	Yarn 3	Yarn 4
Yarn 1	N/A	Beside	Below	Above
Yarn 2	Beside	N/A	Above	Below
Yarn 3	Above	Below	N/A	Beside
Yarn 4	Below	Above	Beside	N/A

Abbildung 2.2-15: Kontaktart [21] S. 154

Um Kollisionen und Schnitte zwischen den Garnen erkennen zu können, wird für bestimmte Punkte an der Oberfläche eines Garns geprüft, ob diese innerhalb eines fremden Garns liegen. Die Prüfung findet hierbei für jeden in Betracht kommenden Garn in zwei Schritten statt. Zunächst wird iterativ die Stelle auf dem fremden Garnpfad mit dem geringsten Abstand zum untersuchten Punkt ermittelt. Hieraus wird der für den untersuchten Punkt relevante Garnquerschnitt ermittelt. Anhand dieses Querschnitts kann nun geprüft werden, ob der Punkt innerhalb oder außerhalb des fremden Garnes liegt. [21] S. 28–30. Für alle Oberflächenknoten wird einzeln geprüft, ob diese innerhalb eines fremden Garnes liegen. Dies geschieht aufgrund der benötigten Rechenkapazität in mehreren Stufen. In der ersten Stufe werden diejenigen Oberflächendreiecke ermittelt, die direkt unter und über dem Knoten liegen. Daraufhin wird durch die Klassifizierungstabelle entschieden ob der Knoten ober oder unterhalb des fremden Garnes liegen sollte. Soll der Knoten oberhalb des Garnes liegen, so wird im geprüft ob der Knoten vor oder hinter dem zuvor bestimmten oberen Dreieck liegt. Die Richtung geht hierbei vom Garnpfad aus. Liegt der Knoten vor dem Dreieck, findet kein Schnitt statt. Liegt der Knoten dahinter, schneiden sich die Garne und üben gegenseitig Druck aufeinander aus. Sollte der Knoten unterhalb des Garnes liegen wird analog das untere Dreieck herangezogen. [21] S. 155–156 (vgl.

Abbildung 2.2-16).

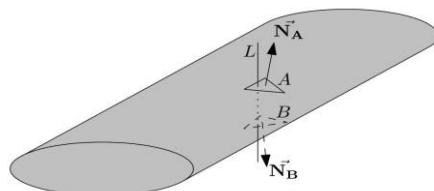


Abbildung 2.2-16: Kreuzung [21] S. 155

Auch für parallel verlaufende Garne findet die Kontakttermittlung für jeden Knoten einzeln statt. Vereinfachend wird angenommen, dass die Garne sich lediglich oberflächlich schneiden. Die „Schnitttiefe“ muss kleiner als die halbe Garnbreite sein. Diese Annahme ist durch den ellipsen-/linsenförmigen Querschnitt aufgrund der typischerweise im Vergleich zur Höhe großen Breite für die mit TexGen bearbeitbaren Modelle zulässig. Es wird nun der minimale Abstand zum fremden Garnpfad ermittelt. Ein Schnitt liegt vor, sobald der ermittelte Abstand kleiner als die halbe Querschnittsbreite ist. [21] S. 156–157. Kann eine Kollision zwischen zwei oder mehr Garnen nicht durch Rotation der Garne vermieden werden, wird die Form des Querschnitts angepasst, indem die Referenzpunkte auf der Oberfläche in Richtung des Garnpfades verschoben werden. Bevorzugt angewandt wird jedoch eine Rotation des Garnquerschnitts wie in Abbildung 2.2-17 erkennbar ist [21] S. 55.



Abbildung 2.2-17: Verformung durch Rotation [21] S. 55

Der Rotationswinkel des Garnes an der betrachteten Stelle ergibt sich aus Formel 2-2, wobei h die Garnhöhe und s den Abstand zwischen den Garnpfaden repräsentiert.

$$\theta = \tan^{-1} \left(\frac{h}{2s} \right)$$

Formel 2-2: Rotationsformel Kollisionen [21] S. 55

Um die Kontaktstellen zwischen den Garnen zu lokalisieren, werden die Oberflächen der Garne durch Dreiecke modelliert. [21] S. 153. Die ausgeübten Kräfte werden proportional zu der Tiefe des Einschnitts angenommen. [21] S. 157. Die Verformung und mechanische Modellierung des Garns geschieht mit der FE-Methode bezüglich der Knoten. Aus diesem Grund werden die wirkenden Kontaktkräfte auf eines der Dreiecke auf die Knoten, welche das entsprechende Dreieck definieren, aufgeteilt. [21] S. 158. Als Input nimmt TexGen lediglich die Garnattribute und die gewünschten Ausmaße des Modells auf [21] S. 7. Ausgegeben werden Dateien der Typen IGES und STEP. Es handelt sich hierbei um frei verfügbare Dateitypen, die von einer Vielzahl an CAD-Programmen weiterverarbeitet werden können. [21] S. 31. Um die Qualität und Integrität der erzeugten Daten sicherzustellen, müssen bestimmte Anforderungen an das Modell erfüllt werden. Der Kontaktalgorithmus funktioniert beispielsweise nur bei ebenen Geflechten. Somit ist eine Anwendung von TexGen auf dreidimensional geflochtene Textilien nicht möglich [21] S. 153. Außerdem müssen die Garnoberflächen über ausreichend viele Knoten diskretisiert werden, um eine genaue Kollisionserkennung sicherzustellen [21] S. 30. Einschränkend wirkt sich hierbei der aufzuwendende Rechenaufwand für die Diskreditierung der Modelle in Voxel aus. In einem Test benötigten beispielsweise 800.000 Voxel 5 Tage auf einem 32 CPU System von 2014. [25] S. 289. Insbesondere bei der Simulation von Geflechten wirkt sich die Annahme von homogenen Garnen negativ aus, da die Scherkräfte auf die einzelnen Fasern nicht exakt abgebildet werden können. [21] S. 139. In zukünftigen Versionen von TexGen können die Oberflächenknoten auch auf Positionen außerhalb des Referenzrahmens der Einheitszelle verschoben werden. [21] S. 163. Außerdem sollen zukünftig auch mehr Textiltypen von TexGen verarbeitet werden können [21] S. 33.

Kommentiert [CD48R47]:

2.3 MODELLIERUNG

Die Modellierung von Faserverbundbauteilen in allen genannten Softwarelösungen erfolgt indem die gesamte vorhandene Geometrie zunächst in Einheitszellen zerlegt wird. Durch das Entfernen des Harzes werden die Einheitszellen zu reinen textilen Geflechten. Innerhalb dieser Geflechte werden wiederum einzelne Fasern betrachtet. Die Fasern werden durch eine Vielzahl an Filamenten beschrieben. Diese Filamente bilden die elementaren Objekte des Modells. Innerhalb dieses hierarchischen Modells vererben die unteren Hierarchieebenen ihre Attribute an die Oberen. Die Filamente enthalten die folgenden Attribute: Dichte, Durchmesser, Elastizitätsmodul, Querkontraktionszahl, Torsionsmodul,

maximale Dehnung und Reißlänge. Neben den durch die Filamente bestimmten Eigenschaften wird durch die Faser die Art der Faser (z.B. Einzelfilament oder Multifilament), die Form des Faserquerschnitts, die Reibkoeffizienten, das Biegeverhalten und die vorliegende Komprimierung der gesamten Filamente bestimmt. Die nächsthöhere Ebene der Geflechte gibt das Verhalten unter Krafteinflüssen in Bezug auf die Faserverteilung und Form wieder. Durch das Hinzunehmen der Matrix zu den reinen textilen Geflechten entsteht eine sogenannte Einheitszelle. Diese bildet nicht nur die mechanischen Eigenschaften unter einer anliegenden Belastung, sondern auch die Permeabilität der jeweiligen Zelle ab. Aus der Gesamtmenge der Einheitszellen ergibt sich schließlich das endgültige Bauteil. Innerhalb dieses Modells kann sowohl der Harzfluss bei der Injektion als auch das Verhalten im Umformprozess und unter wirkenden Kräften und Momenten angegeben werden. Der gesamte Aufbau wird in Abbildung 2.3-1 visualisiert [26].

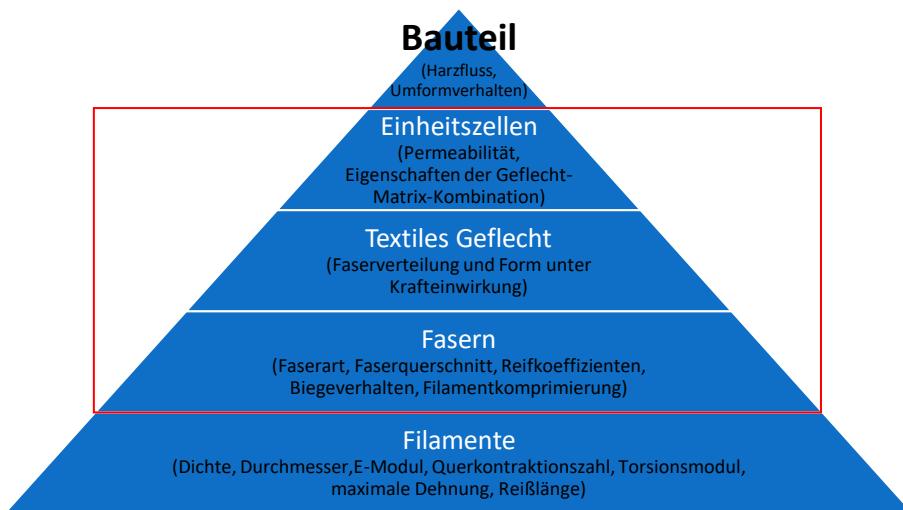


Abbildung 2.3-1 Aufbau virtuelles Modell

Aufgrund ihrer Größe werden die Filamente der mikroskopischen Ebene und das komplette Bauteil der makroskopischen Ebene zugeordnet. Typische Dimensionen der drei Ebenen werden in

Abbildung 2.3-2 dargestellt.

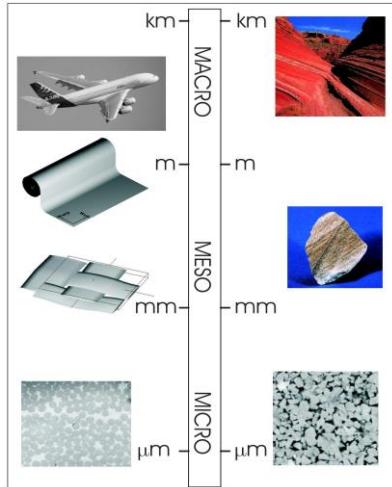


Abbildung 2.3-2: Dimensionen der verschiedenen Modellierungsebenen [27] S. 3

Die vorliegende Arbeit beschränkt sich lediglich auf die mesoskopischen Ebenen (rot markiert in Abbildung 2.3-1), da die Gesamtheit der Filamente als homogen angenommen wird, und das Bauteilverhalten maßgeblich durch die jeweiligen Einheitszellen bestimmt wird [13] [21] [28]. Das Mesomodell soll die geometrischen Abbildungen der Garne, deren Faservolumenanteile und die Randbedingungen der Garne untereinander, wie Reibwerte enthalten [13] S. 1871. Die Eigenschaften der im RTM-Verfahren hergestellten Bauteile werden maßgeblich von dem Fließverhalten der Matrix, ergo der Permeabilität der Textilien, bestimmt. Um zeit- und kostenintensive experimentelle Testreihen zu vermeiden, kann die Permeabilitätsanalyse ebenfalls mittels der digitalen Modellierung auf Meso-Ebene durch das geometrische Modell vorhergesagt werden (vgl. Abbildung 2.3-3). [24] S. 8.[24] S. 10.

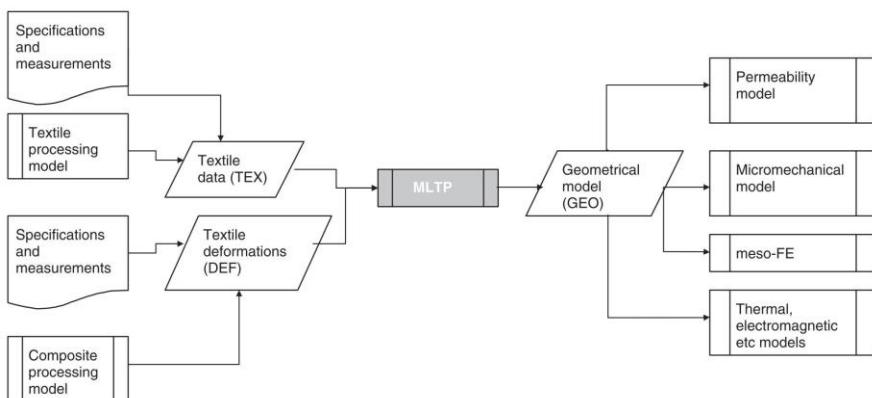


Abbildung 2.3-3: Workflow Fasersimulation [24] S. 3

2.4 VOXELERZUGUNG

Für die FE- und Permeabilitätsanalyse wird für jeden Punkt im untersuchten Raum anhand der zuvor ermittelten Garnverläufe entschieden, ob dieser inner-, bzw. außerhalb eines Garnes liegt. Aus den gleichmäßig über das untersuchte Volumen verteilten Punkten werden Voxel abgeleitet, indem die Punkte anhand ihrer Lage einem Volumenelement zugeordnet werden (siehe Abbildung 2.4-1).

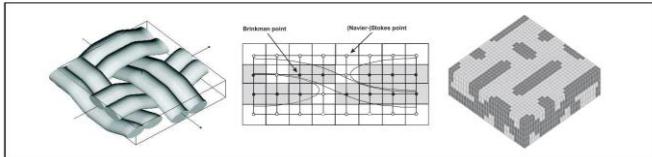


Abbildung 2.4-1: Prozess Voxelerzeugung [27] S. 8

Für alle innerhalb eines Garnes liegenden Voxel werden die Fasereigenschaften angewandt. Die Eigenschaften von Punkten, die außerhalb der Garne liegen, bleiben undefiniert. [15] S. 60. Die mechanischen Eigenschaften der Faservoxel werden durch deren Faservolumenanteil und die durchschnittliche Faserrichtung bestimmt. Innerhalb der Voxel wird die Struktur daraufhin als homogen betrachtet. Es wird hierfür die Formel 2-1 verwendet. V beschreibt das Voxelvolumen, v_f den Faseranteil und a_f die Faserrichtung am untersuchten Punkt.

$$V_f = \frac{1}{V} \int v_f dv, \quad \mathbf{A}_f = \frac{1}{V} \int \mathbf{a}_f dv$$

Formel 2-3: Voxelhomogenisierung [15] S. 60

Die Anzahl der Voxel, und somit der Diskretisierungsgrad wird vom Nutzer festgelegt. [15] S. 59. Für alle Punkte wird entschieden ob diese innerhalb oder außerhalb der Rovinge liegen. Würfel die von den Rovingen lediglich geschnitten werden, werden je nach Anteil der Fasern weiterverarbeitet. [27] S. 8.

2.5 QUERSCHNITTSVERFORMUNG

Die Form der Garnquerschnitte kann nur schwer vorhergesagt werden, da diese einer Vielzahl an Einflüssen unterliegen. [29] S. 2435. Nichtsdestotrotz gibt es einige Versuche die Querschnittsverformungen von Rovingen zu simulieren. Im folgenden Abschnitt werden die Arbeiten von Sherburn (2007) und P.Badel et al. (2008) Zu diesem Thema vorgestellt. Insbesondere letztere entwickelten einen Algorithmus zur ausreichend genauen Querschnittsvorhersage wie in Abbildung 2.5-1 ersichtlich wird und im Folgenden vorgestellt wird. Die Untersuchungsergebnisse basieren jeweils auf Röntgenmikroskopaufnahmen.

Kommentiert [CD58R57]: Quelle

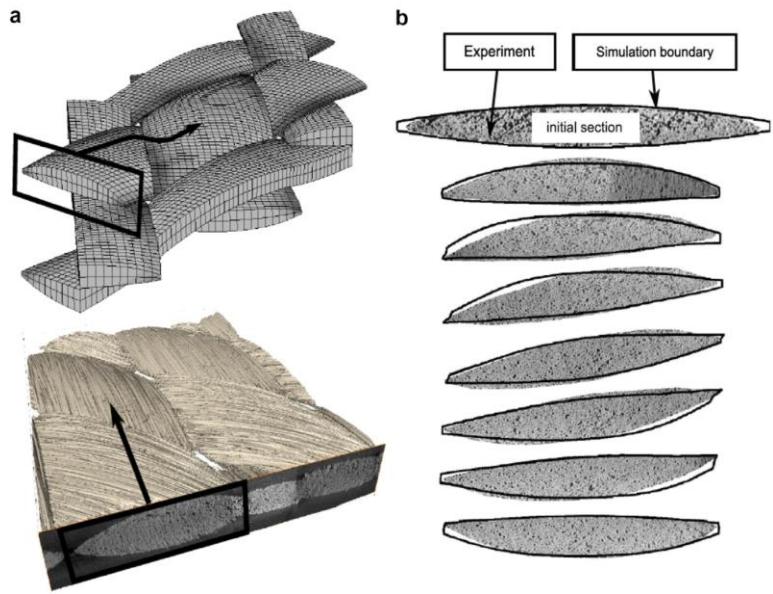


Abbildung 2.5-1: Vergleich realer und simulierter Querschnitte [29] S. 2439

Der Algorithmus basiert auf der Annahme, dass alle Kollisionen durch Rotationen der Querschnittsflächen verhindert werden können. Es zeigt sich in den Abbildungen deutlich, dass sich der Garnquerschnitt an die Kontaktflächen anschmiegt. Dieses Verhalten lässt sich auch in Abbildung 2.5-2 von Sherburn erkennen.

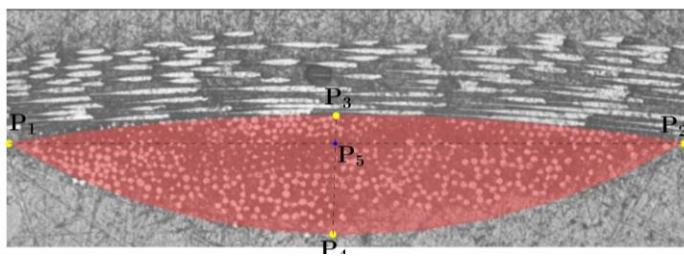


Abbildung 2.5-2: Beschreibung Querschnitt [21] S. 49

3 Untersuchung

3.1 GENERELLES

3.1.1.1 Beschreibung des reduzierten Modells

Die vorliegende Arbeit basiert auf einem eingeschränkten „Full-Data-Voxel“-Modell. Jedem Voxel wird neben seiner Position im Mesh, seine Rovingzugehörigkeit und die Faserrichtung zugeteilt. Das Modell wird insofern minimiert, als dass alle wirkenden Belastungen nicht weiter betrachtet werden. Es wird somit lediglich ein geometrisches Modell mit wenigen zusätzlichen Metadaten erzeugt (vgl. 3.1.1.3). Die Verformung aufgrund von Kollisionen erfolgt deshalb durch einen experimentell ermittelten Algorithmus (siehe 3.2.4).

3.1.1.2 Voxel

Der Begriff Voxel beschreibt ein dreidimensionalen Pixel mit festgelegten X-, Y-, Z-Koordinaten [30]. Das Programm nutzt als Referenzobjekt gleichseitige Würfel, deren Kantenlänge in der Datei Einstellungen vom Nutzer festgelegt wird. Die Position der einzelnen Würfel wird jeweils durch diejenige Ecke bestimmt, welche über den maximalen Wert in X-, Y- und Z-Richtung verfügt (vgl. Abbildung 3.1-1)

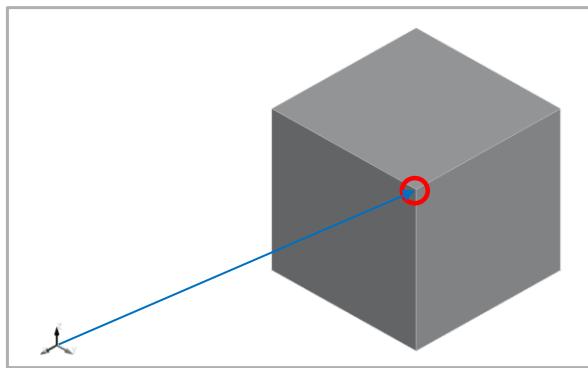


Abbildung 3.1-1 Ortsvektor zur Bestimmung der Voxelposition

3.1.1.3 Voxelliste

Die zwischen den einzelnen Programmmodulen ausgetauschten Daten werden in einer Tabelle (Tabelle 3.1-1) mit zehn Spalten angeordnet, wobei jeder Voxel über eine Zeile verfügt.

Voxel ID	X-Position	Y-Position	Z-Position	Eindeutiger Positions Index	Part-ID	Shell-ID	Voxel-typ	Kollisions-koerper	Kollisions-indikator
1	12	2	23	22	1	12	1	0	0
2	13	543	23	34	1	546	2	4	1

Tabelle 3.1-1 Liste der Voxel

Die Spalte „Voxel ID“ weist jedem Voxel eine eindeutige Identifikationsnummer zu. Die Werte in den Spalten „X-, Y-, Z-Position“ enthalten die jeweils entsprechenden Koordinaten des Voxels in dem zuvor eingeführten infiniten Gitter relativ zu dem vom Programm bestimmten Gitterursprung. Die Spalte 5 „Eindeutiger Positionsindex“ erstellt aus den X-, Y-, und Z-Anteilen des Ortsvektors einen eindeutigen Index zur Verbesserung der Laufzeit (Siehe Indexierung 3.5.1.1.3 3.5.1.1.5). Die Spalte „Part-ID“ weit den Voxel einem Roving (Part-ID > 1) oder der Kavität (Part-ID = 1) zu. Die Shell-ID in Spalte 7 verknüpft jeden Voxel mit der Shell, über welche er extrudiert wurde (vgl. 3.2.2). Die Einträge in Spalte 8 „Voxeltyp“ werden in der folgenden Tabelle 3.1-2 erläutert.

Eintrag	0	1	2	5	6
Bedeutung	Dummy-Voxel	Innenliegendes Voxel	Voxel am Rand des jeweiligen Parts	Inlet-Voxel der Kavität	Outlet-Voxel der Kavität

Tabelle 3.1-2 Liste der Voxeltypen

In Spalte 9 wird der Kollisionskörper (siehe 3.2.4.1.3) hinterlegt und die Spalte 10 gibt an ob der Voxel sich die Position im Mesh (vgl. 3.1.1.4) mit einem beliebigen anderen Voxel teilt.

3.1.1.4 Geordnetes Gitter

Die Würfel werden in einem vom Programm erstellten, virtuellen, infiniten Gitter positioniert. Die Diskretisierung des Gitters stimmt mir der Kantenlänge der Voxelkanten überein. Hierdurch ist die Position aller Voxel stets ganzzahlig. Zur Nutzung der unter 3.5.1.1 vorgestellten Methode wird der Ursprung des virtuellen Gitters so positioniert, dass alle Positionsangaben der Voxel positiv sind.

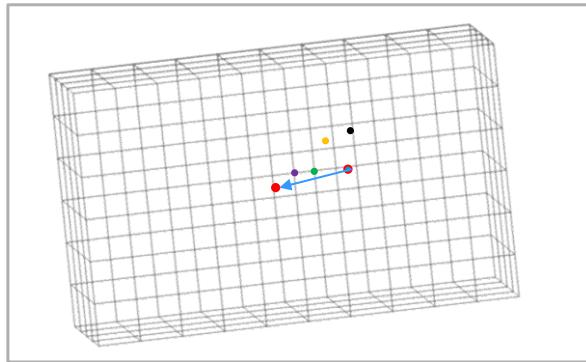


Abbildung 3.1-2 Verschiebung des Gitterursprungs

Hinweis: Die Begrenzung des Gitters in Abbildung 3.1-2 erfolgt aus darstellungstechnischen Gründen.

3.1.1.5 Anwendungseinstellungen

Der Nutzer spezifiziert in der Funktion „Einstellungen“ die zu importierenden Dateien mit den Faser- bzw. Kavitätsdaten, legt die Ausgabedatei des Programmes fest, und trifft die folgenden Einstellungen:

- Faserquerschnittshöhe**
Die Faserquerschnittshöhe gibt die für das Programm zu verwendende Höhe des Faserquerschnitts an. Die Eingabe erfolgt hierbei in der Einheit Millimeter. Die Funktion „Einstellungen“ runden den eingegebenen Wert automatisch in ein Vielfaches der Voxelkantenlänge.

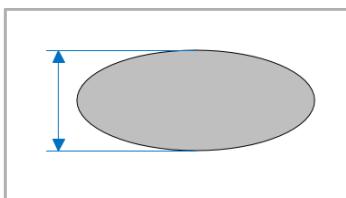


Abbildung 3.1-3 Skizze Rovinghöhe

- Kantenlänge der Voxel**

Durch diesen Eintrag wird die Auflösung des Gitters und somit die Qualität der Ergebnisse festgelegt. Durch das Format der Ausgabedatei ist die maximale Voxelanzahl auf 100 Mio begrenzt. Zwischen 12,5 Mio 100 Mio Voxeln kann es unter Umständen zu Fehlern kommen.

Es empfiehlt sich daher die Kantenlänge der Voxel an das gemeinsame maximale Volumen der Rovinge und der Kavität anzupassen. Die Erhöhung der Auflösung, respektive Verkleinerung der Voxelkantenlänge beeinflusst die Laufzeit gemäß Abschnitt 3.5.1.1.1 maßgeblich.

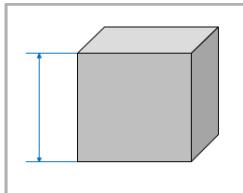


Abbildung 3.1-4 Skizze Kantenlänge Voxel

- **Injektionsrichtung**

Die Angabe der Harzinjektionsrichtung definiert die Verteilung der In- und Outlets der einzelnen Parts. Die Auswahl erfolgt über die Eingabe der folgenden Zahlen in dem entsprechenden Feld:

für die X- Richtung	für die Y- Richtung	für die Z- Richtung
2	3	4

Tabelle 3.1-3 Auswahl Injektionsrichtung

3.1.2 Prinzipien

3.1.2.1.1 Modularität

Der Programmcode ist modular aufgebaut um auch komplexe Funktionen übersichtlich darzustellen. Hierdurch wird die Wartbarkeit und künftige Erweiterbarkeit des Programmes sichergestellt [31]. Funktionsnamen, welche von lediglich einer anderen Funktion aufgerufen werden beginnen mit einem **F...**, während die Bezeichnungen von Funktionen mit verschiedenen Mutterfunktionen mit **HF...** beginnen. Die Zahlenfolge hinter dem F verdeutlicht die Zugehörigkeit der jeweiligen Funktion zu den jeweiligen übergeordneten Funktionen.

3.1.2.1.2 Parallelisierung

Die Verarbeitung der Voxeldaten erfordert häufig die Anwendung einfacher Rechenoperationen auf eine große Anzahl an Objekten. Durch die Nutzung von parallelisierten Funktionen der Matlabfunktionsbibliothek kann die Laufzeit für diese Anwendungsfälle auf zeitgemäßen Prozessoren entscheidend verbessert werden (vgl. 3.5.1.1.6) [32].

3.1.2.1.3 Nutzung bestehender Funktionen

Zur Minimierung des Programmieraufwandes greift das vorliegende Programm auf eine maximale Anzahl an Funktionen der Matlab eigenen Programmzbibliothek zu. Durch deren ausführliche Dokumentation in der Matlabhilfe unterstützt dieses Vorgehen die Anforderungen an die künftige Wartbarkeit und Erweiterbarkeit. Des Weiteren sind die Funktionen bereits bzgl. Geschwindigkeit und Ressourcenverbrauch hin optimiert, und gewährleisten somit eine stabile und schnelle Programmausführung.

3.1.2.1.4 Minimierung der Anzahl der Suchfunktionen

Mit dem abermaligen Ziel der Laufzeitverkürzung. Wurde im Laufe der Entwicklung die Anzahl der genutzten Suchfunktionen systematisch verringert. Die Suchfunktionen wurden hierfür durch die Nutzung von indexierten Variablen oder durch Funktionen mit expliziten Programmanweisungen (vgl. 3.5.1.1.5 und 3.5.1.1.2) substituiert.

3.2 VERARBEITUNG DER FASERDATEN

3.2.1.1 Funktionsaufbau

Die Verarbeitung der Faserdaten geschieht in insgesamt fünf Schritten. Zunächst werden die vom Nutzer ausgewählten Dateien vom Programm in von Matlab lesbare Matrizen eingelesen. Die Struktur der

Daten bleibt dabei weitestgehend erhalten (vgl. 3.2.1.1.2). Für die betrachtete Einheitszelle ergibt sich daraus die Abbildung 3.2-1.

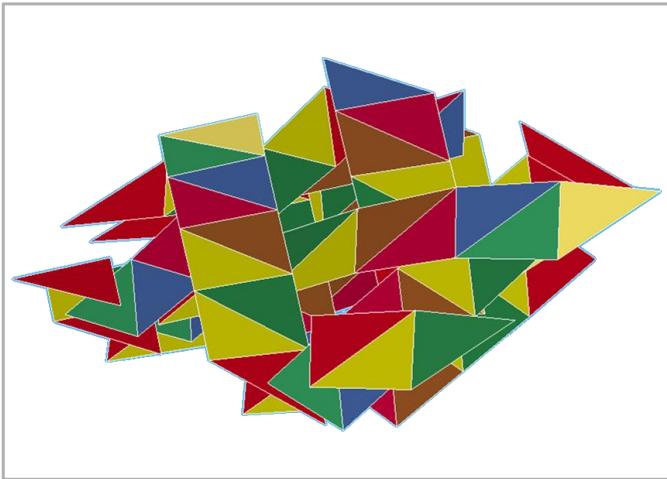


Abbildung 3.2-1 Visualisierte Eingabedaten

Zu beachten ist hierbei, dass die einzelnen Shells noch als eigenständige Einzelteile durch das Programm gehandhabt werden. Da die Verarbeitung der Faserdaten nach Rovingen getrennt geschieht werden die eingelesenen Shells im nächsten Schritt den Rovingen zugeordnet. Die Zugehörigkeiten werden hierbei über gemeinsame Knoten ermittelt (vgl. 3.2.1.1.3). Des Weiteren werden den Shells und Knoten innerhalb derselben Funktion ihre Reihenfolge innerhalb des Rovings zugewiesen. Hieraus ergibt sich abschließend das in Abbildung 3.2-2 dargestellte Geflecht.

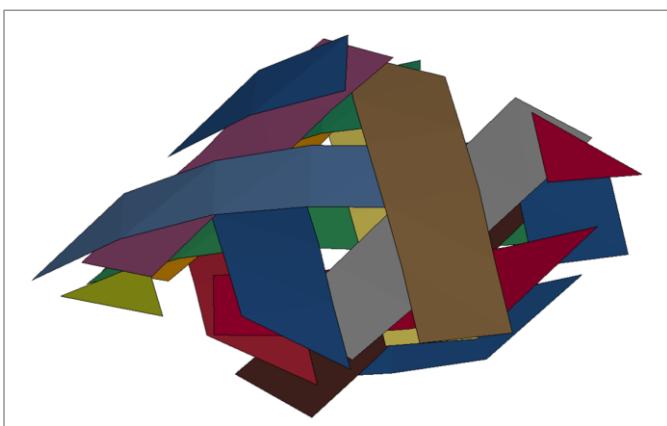


Abbildung 3.2-2 Visualisierte aufbereitete Eingabedaten

Um den noch flachen Rovingbändern (Höhe = 0mm) einen realistischen Querschnitt zuzuweisen, wird über die Grundfläche jeder Shell eine Ellipse in Faserrichtung extrudiert (siehe 3.2.2). Es ergibt sich hieraus der in Abbildung 3.2-3 dargestellte Zustand. Dieses Verfahren bildet im Wesentlichen das Verfahren von Lomov [17] nach, mit dem Unterschied, dass die einzelnen Volumen als Voxelmenge diskretisiert werden und durch die doppelte Angabe der Faserrichtung (rechter und linker Rand des Rovings) eine Verdrillung des Rovings direkt abgebildet wird.

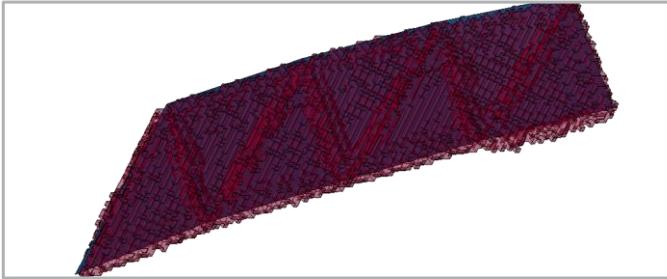


Abbildung 3.2-3 Extrudierter Roving in Draufsicht inkl. Shells

Die Übergänge zwischen den extrudierten Shells innerhalb der Rovings werden durch die darauffolgende Funktion (3.2.3) hergestellt, um den Rovingquerschnitt über die Faserrichtung konstant zu halten und Spalte zu vermeiden. Die genannten Probleme sind in Abbildung 3.2-4 in der Seiten- und Querschnittsansicht in Abbildung 3.2-5 zu erkennen.

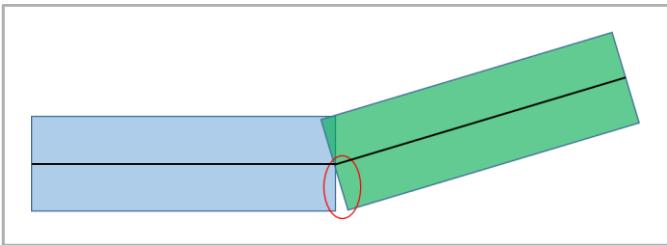


Abbildung 3.2-4 Darstellung Übergangsproblematik Seitenansicht

An der Abbildung 3.2-4 ist zu sehen, dass es bedingt durch die Faserrichtungsänderung und dem von den jeweiligen Vorgänger- und Nachfolger-Shells unabhängigen Extrusionsprozess an der Anstoßkante zu Spaltbildung und Verdichtung kommt.

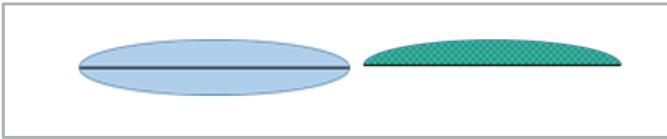


Abbildung 3.2-5 Darstellung Übergangsproblematik Querschnitt

Die linke Darstellung in Abbildung 3.2-5 stellt hierbei den Rovingquerschnitt außerhalb des Übergangsbereiches dar. Die Rechte hingegen veranschaulicht, dass die Querschnittsfläche in dem Übergangsbereich halbiert wird. Die noch besetzten Meshpositionen sind jedoch zweifach belegt. Nach der Übergangsfunktion ergibt sich für einen Beispielroving die Abbildung 3.2-6. Die von der Funktion verarbeiteten Voxel sind in der Visualisierung blau hervorgehoben.

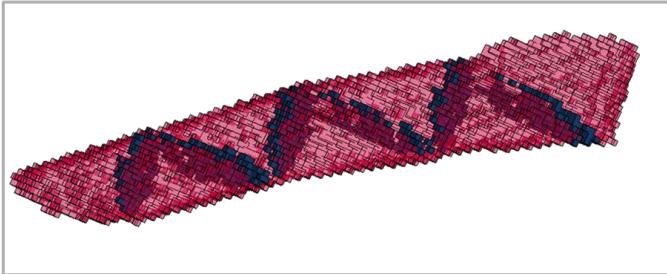


Abbildung 3.2-6 Darstellung durch die Übergangsfunktion verschobener Voxel

Zur Sicherstellung eines konstanten Faservolumens werden abschließend die vorkommenden Kollisionen zwischen einzelnen Voxel unterschiedlicher Rovinge durch die Umformfunktion beseitigt. Dies geschieht in mehreren Schritten, in welchen die betroffenen kollidierenden Voxel, auf freie Mesh-Positionen verschoben werden und somit die Querschnittsform des Rovings angepasst wird (siehe 3.2.4). In Abbildung 3.2-7 werden die hierdurch verschobenen Voxel in blauer Farbe hervorgehoben.

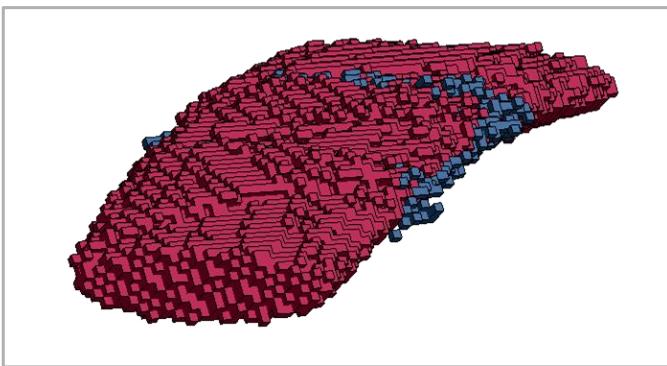


Abbildung 3.2-7 Darstellung durch die Umformfunktion verschobener Voxel

3.2.1.1.2 Einlesen der Eingabedaten

Die zu importierenden Rovingdaten liegen in einer „.pc“-Datei vor. Die Dateistruktur entspricht dem folgenden Muster:

- Die aus der Flechtsimulation entstammenden Faserbänder sind durch aneinander gereihte Dreiecke diskretisiert (vgl. Abbildung 3.2-10).
- Die Eckpunkte der Dreiecke verfügen über eine eindeutige, positive sogenannte Knoten-Identifikationsnummer. Jeder Identifikationsnummer, im folgenden ID abgekürzt, werden die Koordinaten des Knotens in X-, Y- und Z-Richtung zugewiesen. Die Angabe erfolgt in Millimeter und der Ursprung des verwendeten Koordinatensystems entspricht demjenigen der Flechtprozesssimulation. Jeder Knoten verfügt in der Datei über eine Zeile und wird gemäß der Abbildung 3.2-8 und Abbildung 3.2-9 in der Datei beschrieben.

S#	IDNOD	X	Y	Z
NODE /	35619.4299488067626432.5873564513573-73.472165743865			
NODE /	35629.3722667694091432.6415671141991-75.465451876678			

Abbildung 3.2-8 Notation Knoten Input

- Die Zeile beginnt mit dem Schlüsselwort „NODE“. In den Spalten 9 bis 16 steht die 8-stellige Knoten-ID, gefolgt von den jeweils 16-stelligen X-, Y- und Z-Koordinaten.
Auf die Liste der Knoten folgt eine Auflistung aller vorkommenden Dreiecksflächen. Diese werden im Folgenden als Shells bezeichnet. Jede Shell wird wiederum über eine Zeile beschrieben.

\$#	IDEL	IDPRT	IDNOD1	IDNOD2	IDNOD3	IDNOD4
SHELL /	3554	2000	3561	3562	51850	0
SHELL /	3555	2000	3562	3563	51851	0

Abbildung 3.2-9 Notation Shell Input

- Die Zeile beginnt mit dem Schlüsselwort „SHELL“. Wie bereits die Knoten, verfügt auch jede Shell über eine eindeutige positive Identifikationsnummer, welche sich in den Spalten 9 bis 16 befindet. Die IDs der referenzierten Knoten können den Zeilen jeweils ab Spalte 25 entnommen werden. Sowohl die hier referenzierten Knoten- als auch die Shell-IDs sind 8-stellig. Wie in der Abbildung 3.2-9 zu erkennen ist, ist die Dateistruktur auf viereckige Shells ausgelegt. Im Rahmen der Faserdateninput-funktion werden jedoch lediglich Dreiecke verarbeitet, weshalb der vierte referenzierte Knoten die ID Null zugewiesen bekommt.
Die Einlesefunktion erzeugt aus den vorliegenden Daten eine Knoten- und eine Shellmatrix, in die die Shell- und Knotenliste der Eingabedatei kopiert werden.

3.2.1.1.3 Trennen der Rovinge

Die Funktion zur Zuordnung der einzelnen Shells zu den entsprechenden Rovingen wird im Folgenden beispielhaft für lediglich einen Roving erläutert. Die Ermittlung der weiteren Rovingzugehörigkeiten geschieht analog bis alle eingelesenen Dreiecke und Knoten einem Roving zugeordnet wurden. Zunächst wird für alle Knoten-IDs ermittelt, wie oft jeder Einzelne Knoten in verschiedenen Shells vorkommt, die noch keinem Roving zugordnet wurden (z.B. Tabelle 3.2-1). Als ersten Knoten eines jeden Rovings wird ein Knoten gewählt, der lediglich in einer noch nicht zugeordneten Shell vorkommt.

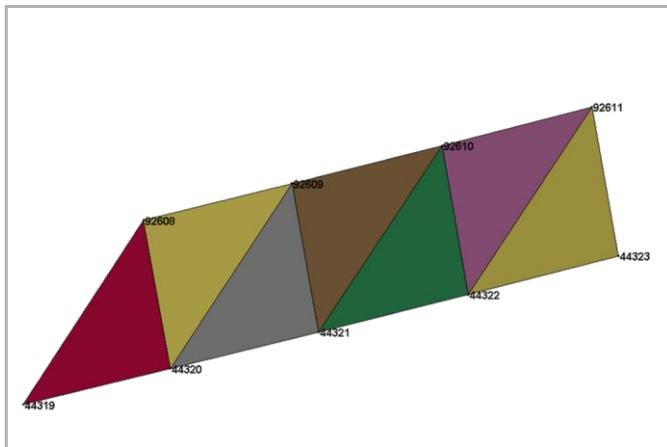


Abbildung 3.2-10 Darstellung eines einzelnen Rovings inkl. Knoten-IDs

In diesem Beispiel verfügt der ausgewählte Knoten über die ID 44319. Die den Knoten beherbergende Shell wird dem aktuellen Roving Index zugeordnet und wird als erste Shell innerhalb des Rovings markiert. Wie im Vergleich zwischen der Tabelle 3.2-1 und Tabelle 3.2-2 zu erkennen ist, wird die Anzahl aller Knoten, welche Teil der soeben zugeordneten Shell sind, um eins verringert. Als nächster Knoten wird derjenige ausgewählt der Bestandteil der vorhergehenden Shell ist und lediglich in einer noch nicht zugeordneten Shell vorkommt (in diesem Beispiel der Knoten mit der ID 92608). Diese

Vorgehensweise lässt sich bis zur letzten Shell fortsetzen. In der letzten Shell eines Rovings kommt es jedoch zu dem Sonderfall, dass alle Knoten der Shell gleichzeitig die Anzahl eins aufweisen. Die Knotenreihenfolge innerhalb dieser Shell wird deshalb über eine spezielle Funktion separat bestimmt. Hierbei wird wiederum die Anzahl der Vorkommnisse der Knoten-IDs in allen eingelesenen Shells ermittelt. Der dreimalig vorkommende Knoten wird vor dem zweimalig bzw. einmalig vorkommenden Knoten platziert.

Knoten ID	Anzahl in nicht zugeordneten Shells	Knoten Reihenfolge
44319	1	Unklar
92608	2	Unklar
44320	3	Unklar
92609	3	Unklar
44321	3	unklar
92610	3	Unklar
44322	3	Unklar
92611	2	Unklar
44323	1	Unklar

Knoten ID 1	Knoten ID 2	Knoten ID 3	Shell Reihenfolge
44319	92608	44320	Unklar
92608	44320	92609	Unklar
44320	92609	44321	Unklar
92609	44321	92610	Unklar
44321	92610	44322	Unklar
92610	44322	92611	Unklar
44322	92611	44323	Unklar

Shells

Tabelle 3.2-1 Ursprüngliche Übersicht des Knoten- und Shellstatus

Knoten ID	Anzahl in nicht zugeordneten Shells	Knoten Reihenfolge
44319	0	1
92608	1	Unklar
44320	2	Unklar
92609	3	Unklar
44321	3	unklar
92610	3	Unklar
44322	3	Unklar
92611	2	Unklar
44323	1	Unklar

Knoten ID 1	Knoten ID 2	Knoten ID 3	Shell Reihenfolge
44319	92608	44320	1
92608	44320	92609	Unklar
44320	92609	44321	Unklar
92609	44321	92610	Unklar
44321	92610	44322	Unklar
92610	44322	92611	Unklar
44322	92611	44323	Unklar

Shells

Tabelle 3.2-2 Übersicht Knoten- und Shellstatus nach erfolgter Shellzuordnung

Für den verwendeten Beispielroving ergibt sich abschließend die in Abbildung 3.2-11 dargestellte Struktur. Alle Knoten und Shells wurden sowohl einem Roving als auch einer Reihenfolge innerhalb des entsprechenden Rovings zugeordnet. Benötigt werden die Reihenfolge der Knoten bzw. Shells im weiteren Verlauf der Programmausführung in den Funktionen „Extrudieren“ (vgl. 3.2.2) und „Erstellung der Übergänge“ (siehe 3.2.3).

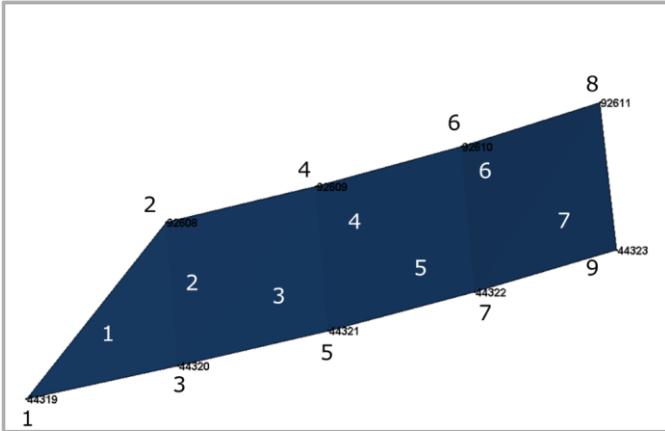


Abbildung 3.2-11 Darstellung eines einzelnen Rovings inkl. Shell (weiß) und Knotenreihenfolge (schwarz)

3.2.1.4 Erstellung des Referenzgitters

Die Funktion konvertiert zunächst die Knotenkoordinaten in Vielfache der Voxelkantenlänge. Hierzu wird das Ergebnis der Formel 3-1 gerundet.

$$\text{Koordinate in mm} \div \text{Voxelkantenlänge} \frac{\text{mm}}{\text{Kantenlänge}} = \text{Koordinate in Kantenlängen}$$

Formel 3-1 Umrechnungsformel von mm in Voxelkoordinaten

Da im weiteren Programmablauf oftmals aufgrund der in Matlab integrierten Indexierungsfunktion **sub2ind** keine negativen Koordinaten möglich sind, wird ein neuer Ursprung eingeführt und die bestehenden Knotenkoordinaten relativ zu diesem angegeben. Für die Position des neuen Referenzursprungs im globalen Koordinatensystem werden zunächst die maximalen und minimalen Koordinaten aller vorhandenen Knoten ermittelt. Der neue Ursprung ergibt sich aus der Formel 3-2.

$$\text{Minima} - (\text{Maxima} - \text{Minima}) = \text{Referenzursprung}$$

Formel 3-2 Translation Ursprung

In Abbildung 3.2-12 wird die Translation des Ursprungs durch den blauen Pfeil dargestellt und der ermittelte Ursprung hervorgehoben. Für die Indexierung mittels der Matlab eigenen **sub2ind** Funktion werden die Ausmaße eines Referenzgitters benötigt. Die Dimensionen des Referenzgitters werden so gewählt, dass alle folgenden Operationen in jedem Fall innerhalb des beschriebenen Bereiches erfolgen. Rechenoperationen außerhalb des Referenzgitters würden entweder zu invaliden Ergebnissen oder zu einem Programmabsturz führen. In Abbildung 3.2-12 werden die Grenzen dieses Referenzgitters durch die roten Linien beschrieben. Die Ausmaße werden, wie ebenfalls in Abbildung 3.2-12 zu erkennen ist, mittels Formel 3-3 ermittelt.

$$3 \cdot (\text{Maximas} - \text{Minimas}) = \text{Gitterausmaß}$$

Formel 3-3 Berechnung Ausmaße Referenzgitter

Abbildung 3.2-12 Darstellung Referenzgitter inkl. Faserdaten

3.2.2 Extrudieren

3.2.2.1 Funktionsaufbau

Die Extrusion des Rovingsquerschnitts in Faserrichtung über die Grundfläche der einzelnen Shells geschieht sequenziell und unabhängig von der Rovingzugehörigkeit der jeweiligen Shells. Die Bearbeitung der Shells wird in einem Shell spezifischen Koordinatensystem durchgeführt. Generiert werden die Voxel in einer auf die als Grundfläche genutzte Shell abgestimmten 3-dimensionalen Matrix. Zur Minimierung des Arbeitsspeicherbedarfs und zur Laufzeitoptimierung werden die ermittelten Voxel nach der Extrusion innerhalb der Matrix in eine Liste konvertiert (siehe 3.5.1.1.3 und 3.5.1.1.4). Abschließend werden die ermittelten Voxelkoordinaten wieder in das ursprüngliche Koordinatensystem zurücktransformiert und bestehende Hohlräume aufgefüllt.

3.2.2.1.2 Koordinatensystem

Der Ursprung des Shell spezifischen Koordinatensystems entspricht den Minima der von der Shell referenzierten Knoten in X-, Y- und Z-Richtung. Die Orientierung der Achsen ergibt sich wie in Abbildung 3.2-13 dargestellt aus der Reihenfolge der Knoten.

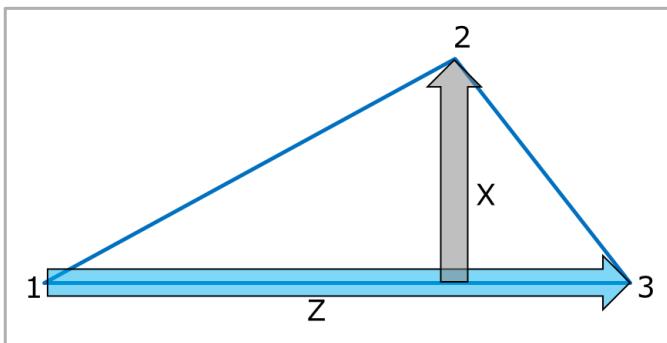


Abbildung 3.2-13 Ableitung des Koordinatensystems aus der Shell

Die Shell selbst bildet hierbei die X-Z-Ebene. Die Z-Richtung verläuft in Faserrichtung (Formel 3-4). Eine vorläufige X-Achse (vgl. Formel 3-5) beschreibt zusammen mit der Z-Achse die Shell-Ebene. Die

Normale zur Ebene ergibt durch Formel 3-6 die Richtung der Y-Achse. Durch die endgültige Bestimmung der X-Achse (Formel 3-7) in der Shell-Ebene und normal zur Z-Achse wird das kartesische Koordinatensystem vervollständigt.

$$v_z = P_3 - P_1$$

Formel 3-4 Bestimmung Z-Achse

$$v_{x,1} = P_2 - P_1$$

Formel 3-5 Bestimmung vorläufige X-Achse

$$v_y = v_z \times v_x$$

Formel 3-6 Bestimmung Y-Achse

$$v_x = v_y \times v_z$$

Formel 3-7 Bestimmung endgültige X-Achse

Für die weitere Verwendung werden die ermittelten Vektoren gemäß Formel 3-8 zu einer Abbildungsmatrix zusammengefasst und normiert.

$$\text{Abbildungsmatrix} = [v_x, v_y, v_z]$$

Formel 3-8 Zusammensetzung Abbildungsmatrix

3.2.2.1.3 Erstellung der Matrix

Wie in Abbildung 3.2-14 zu erkennen ist, werden die Ausmaße der 3-dimensionalen Matrix durch die maximalen Ausmaße der aktuell bearbeiteten Shell und der durch den Nutzer vorgegebenen Rovinghöhe bestimmt.

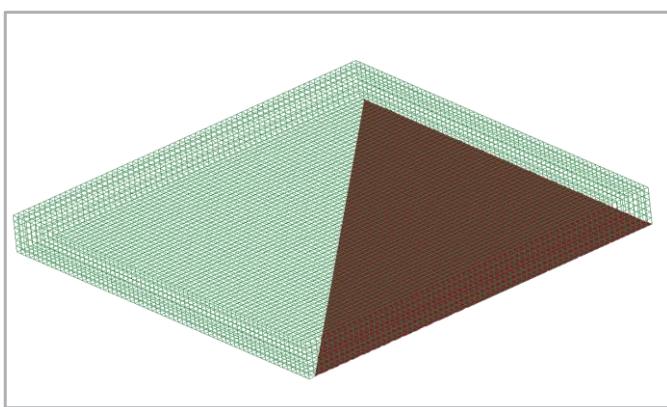


Abbildung 3.2-14 Darstellung der Matrix über der Shell-Grundfläche

Zur Minimierung der Laufzeit der in Abschnitt 8.3.2.2.1.6 vorgestellten Konvertierungsfunktion wird wie in Abbildung 3.2-15 veranschaulicht ist, lediglich eine halbe Ellipse über den Verlauf der Grundfläche extrudiert. Der Querschnitt des Rovings wird für den Flechtprozess als Ellipse angenommen (10).

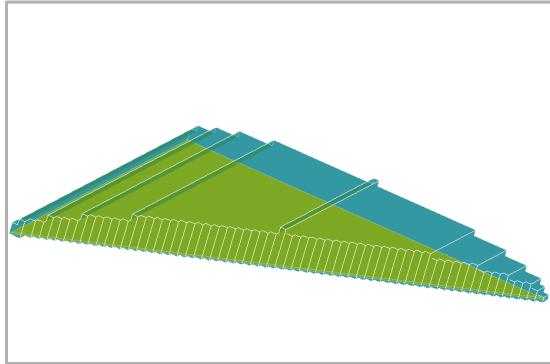


Abbildung 3.2-15 Darstellung der extrudierten Halbellipse über der Shell-Grundfläche

Die Matrix bildet für die Extrusionsfunktion den realen Raum über der Dreiecksgrundfläche ab. Matrixelemente mit dem Wert „1“ stellen vorhandene Voxel dar, während der Wert „0“ innerhalb der Matrix ein unbesetztes Gitterelement repräsentiert. Die die Shellfläche aufspannenden Knotenkoordinaten werden mit der folgenden Formel 3-9 in das sogenannte shellfeste Koordinatensystem transformiert:

$$Koordinaten_{neu} = Abbildungsmatrix^{-1} \cdot (Koordinaten - Ursprung)^T$$

Formel 3-9 Koordinatentransformation

Die Methode dreidimensionale Matrizen zur Raumabstraktion zu nutzen spart aufgrund der inhärenten Verbindung zwischen realer Gitterpositionsbezeichnung und der Position des zugehörigen Speicherorts innerhalb der Matrix Zeit und verringert die Prozessorauslastung durch den Verzicht auf die ansonsten notwendigen Suchfunktionen. Nachteilig wirkt sich die Methode jedoch auf den Arbeitsspeicherbedarf der Funktion aus, da auch unbesetzte Gitterelemente im Arbeitsspeicher gehalten werden müssen (vgl. 3.5.1.1.4). Neben dem Argument der Laufzeitverkürzung spricht auch die einfache Verarbeitbarkeit großer Voxelanzahlen und die geringe Komplexität durch die realitätsnahe Abbildung für den Einsatz von dreidimensionalen Matrizen anstatt von Voxellisten in diesem Szenario.

3.2.2.1.4 Extrusion der Ellipsenform

Die Extrusion der halben Ellipse geschieht zunächst unabhängig von der Form der jeweiligen Shell über die gesamte Gitterfläche in Faserrichtung. Die Faserrichtung entspricht hierbei gemäß Abschnitt 3.2.2.1.2 der Z-Richtung sowohl im Koordinatensystem, als auch in der im vorherigen Abschnitt definierten Raummatrix. Wie in Abbildung 3.2-16 zu erkennen ist, geschieht die Extrusion in Y-Richtung indem zunächst für den jeweiligen X-Wert mittels der Formel 3-10 der Y-Wert ermittelt wird (Schritt 2). Dieser wird daraufhin auf ein Vielfaches der Kantenlänge gerundet (Schritt 3). Abschließend werden alle Gitterelemente zwischen der Grundfläche und dem ermittelten Y-Wert als Voxel gekennzeichnet (Schritt 4).

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1$$

$$\sqrt{b^2 - \left(1 - \frac{(x - x_0)^2}{a^2}\right)} + y_0 = y$$

$$x_0 = \frac{\text{Rovingbreite}}{2}$$

$$y_0 = 0$$

Formel 3-10 Y-Werte der Ellipse Berechnung

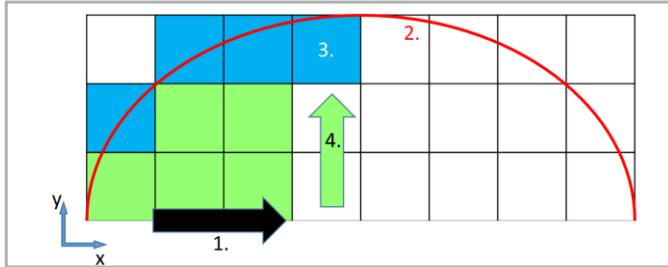


Abbildung 3.2-16 Skizze Extrusionsverfahren

Die Abbildung 3.2-17 stellt den Zustand des Extrusionskörpers über der zugrundeliegenden hellgrünen Shell (vgl. Abbildung 3.2-17) dar. Sowohl die halbe Ellipsenform, als auch die Projektion dieser über die gesamte Matrix in Faserrichtung, sind leicht zu erkennen.

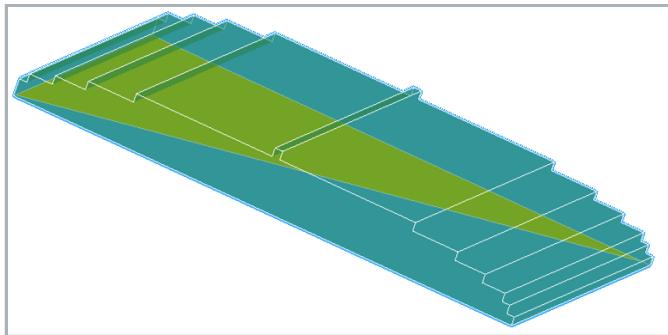


Abbildung 3.2-17 Visualisierung der extrudierten Halbellipse über der Shellfläche

3.2.2.1.5 Anpassung des Extrusionskörpers

Der im vorherigen Schritt erstellte Körper wird nun auf die Dreiecksfläche beschnitten. Der Prozess findet getrennt für die in Abbildung 3.2-18 dargestellten Flächen 1 und 3 statt. Fläche 2 beschreibt weiterhin die Shellfläche gemäß der in Abschnitt 3.2.2.1.2 gewählten Orientierung.

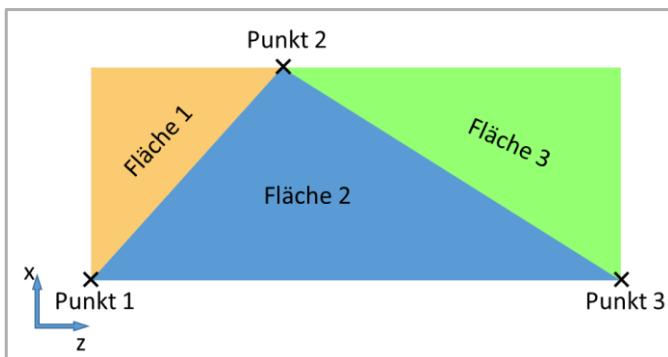


Abbildung 3.2-18 Skizze der Draufsicht auf den Extrusionskörper

Die Entfernung der überschüssigen Voxel geschieht ähnlich zu dem unter 3.2.2.1.4 zuvor vorgestellten Prozess und wird in Abbildung 3.2-19 skizziert. Der Prozess wird für alle vorkommenden X-Werte

sequentiell ausgeführt (Schritt 1). Zunächst wird mittels der Formel 3-11 der zu dem aktuellen X-Wert zugehörige Z-Wert bestimmt (Schritt 2).

$$a(x - x_0) + z_0 = z$$

$$a = \frac{z_{i+1} - z_i}{x_{i+1} - x_i}$$

$$x_0 = x_2, z = z_2$$

Formel 3-11 Berechnung der Geraden zur Beschränkung des Extrusionskörpers

Dieser wird wiederum auf ein ganzzahliges Vielfaches der Voxelkantenlänge gerundet (Schritt 3). Abschließend werden alle Voxel zwischen dem ermittelten Z-Wert und der zugehörigen Matrixrandfläche entfernt, indem den jeweiligen Matrixeinträgen der Wert 0 zugewiesen wird (Schritt 4).

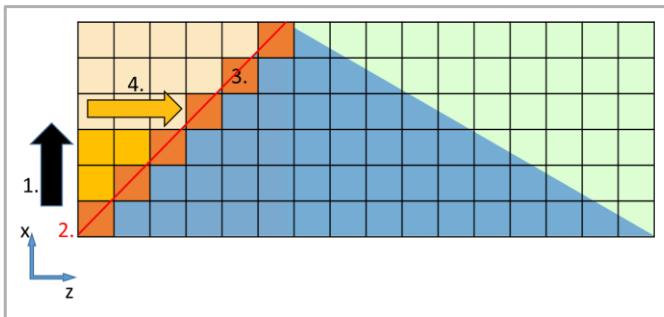


Abbildung 3.2-19 Skizze Beschränkungsverfahren des Extrusionskörpers

Abbildung 3.2-20 stellt den sich ergebenden Körper inkl. der ursprünglichen Shell dar.

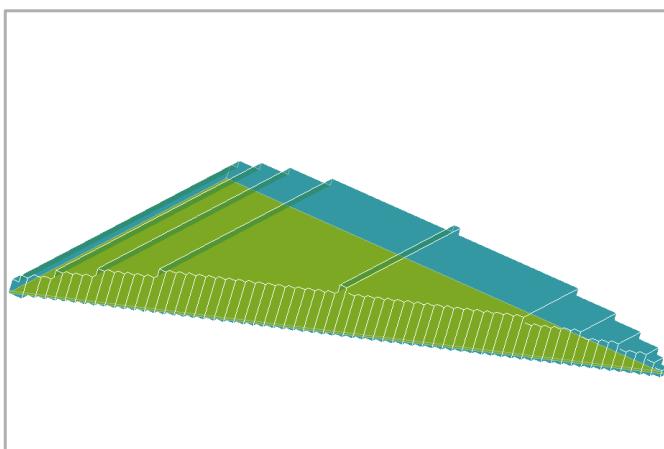


Abbildung 3.2-20 Darstellung des finalen Extrusionskörpers

3.2.2.1.6 Konvertierung der Matrix in eine Liste

Abgeschlossen wird die Extrusionsfunktion durch die bereits erwähnte Konvertierung der Matrix in eine Liste sämtlicher erzeugten Voxel und deren Spiegelung an der Shellebene. Hierzu werden die Positionen aller Matrixelemente mit dem Wert „1“ in der Matrix ermittelt. Diese entsprechen den Ortsvektoren der

Voxel im Shell festen Koordinatensystem (vgl. Voxelliste 3.1.1.1.2). Des Weiteren wird von der Shell die Rovingzugehörigkeit und die Shell-ID übernommen. Tabelle 3.2-3 stellt einige Werte exemplarisch dar.

Voxel ID	X-Position	Y-Position	Z-Position	Eindeutiger Positions Index	Part-ID	Shell-ID	Voxel-typ	Kollisions-koerper	Kollisions-indikator
...
12	2	23			3	12			
13	543	23			3	12			
...

Tabelle 3.2-3 Voxelliste die durch die Extrusion erzeugt wurde

Die Spiegelung erfolgt indem die erzeugte Tabelle kopiert wird und die Y-Positionswerte der Kopie mit -1 multipliziert werden. Anschließend werden die ursprüngliche Tabelle und die Kopie wieder zusammengeführt. Die Voxelkoordinaten werden mit Hilfe von Formel 3-12 in das globale Koordinatensystem zurücktransformiert und die entstandene Liste wird an die bereits bestehende Liste angehängt.

$$Koordinaten_{neu} = Abbildungsmatrix \cdot (Koordinaten - Ursprung)^T$$

Formel 3-12 Rücktransformation in das globale Koordinatensystem

Abschließend ergibt sich für jede eingelesene Shell ein im Folgenden als Extrusionskörper bezeichnetes Volumenelement ähnlich zu dem in Abbildung 3.2-21 dargestellten Körper.

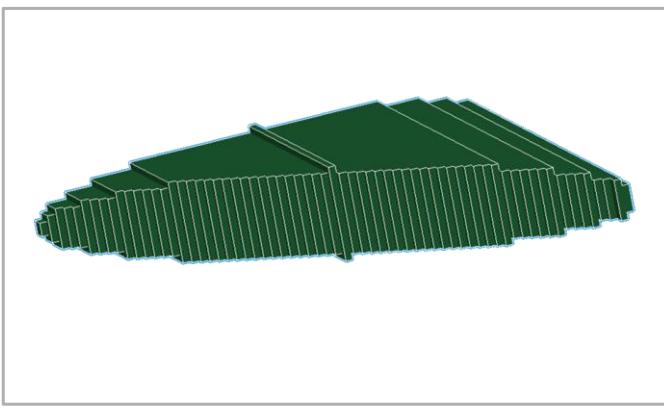
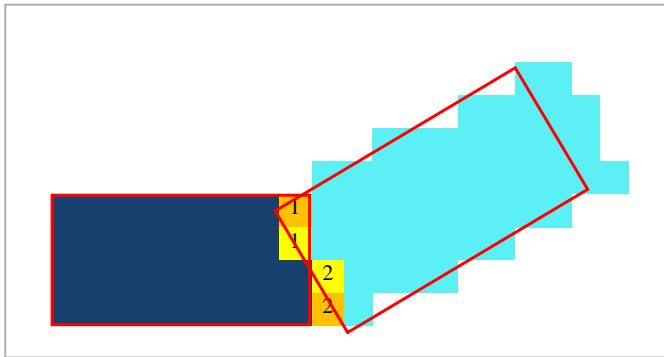


Abbildung 3.2-21 Visualisierung einer Extrudierten Shell

3.2.3 Erstellung der Übergänge

3.2.3.1.1 Funktionsaufbau

Die Funktion verknüpft die extrudierten, noch voneinander unabhängigen Körper mit dem jeweiligen Nachfolger in der Rovingstruktur falls dieser vorhanden ist. Ziel der Funktion ist die Vermeidung der im Abschnitt 3.2.1.1 beschriebenen Risiken bezüglich des Rovingquerschnitts und der Spaltenbildung. Hierfür werden zunächst diejenigen Voxel bestimmt, welche mit dem Nachfolgekörper kollidieren. Daraufhin werden diese an der von den beiden gemeinsamen Punkten definierten Achse gespiegelt (siehe Formel 8.3-13).



Formel 3-13 Darstellung des Vorgehens zur Erstellung der Übergänge

3.2.3.1.2 Kollisionsermittlung

Für die Ermittlung der kollidierenden Voxel wird zunächst allen bisher ermittelten Voxel ein eindeutiger Positionsindex innerhalb des unter Kapitel 3.1.1.4 erzeugten Referenzgitters zugewiesen. Für die Indexierung wird aus Performancegründen die Matlab eigene Funktion `sub2ind` genutzt (siehe 3.5.1.3). Alle Voxel, deren Positionsindex auch im nachfolgenden Extrusionskörper auftaucht, werden nun als kollidierende Voxel zur weiteren Verarbeitung markiert.

3.2.3.1.3 Verschiebung der kollidierenden Voxel

Für die Spiegelung der zu verarbeitenden Voxel werden in einem ersten Schritt zunächst die beiden gemeinsamen Knoten und deren Koordinaten aus der Shell- bzw. Knotenmatrix bestimmt. Für die folgende Funktionsbeschreibung werden die in Abbildung 3.2-22 dargestellten Voxel beispielhaft betrachtet. Die kollidierenden Voxel sind gelb hervorgehoben.

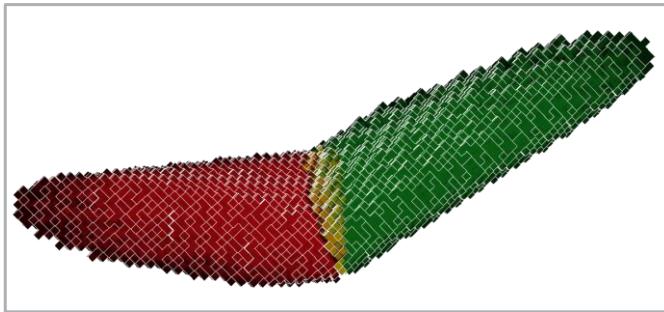


Abbildung 3.2-22 Zustand vor der Spiegelung. Die kollidierenden Voxel sind in Gelb dargestellt

Die Spiegelachse wird definiert durch einen der beiden gemeinsamen Knoten als Ankerpunkt und dem Richtungsvektor V der Formel 3-14 (vgl. Abbildung 3.2-23).

$$v = P_2 - P_1$$

Formel 3-14 Vektor Spiegelachse

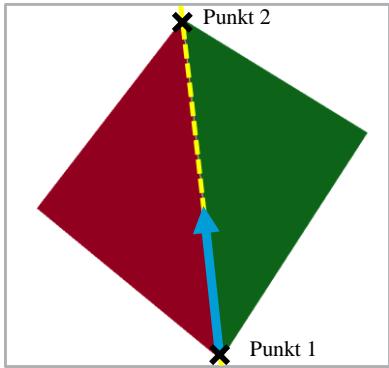


Abbildung 3.2-23 Bestimmung der Spiegelgeraden

Die Bestimmung der neuen Voxelposition geschieht mit Formel 3-15, die sequentiell auf die einzelnen Kollisionsvoxel des betroffenen Extrusionskörpers angewandt wird. Zu diesem Zweck wird zunächst der Lotfußpunkt gemäß der Abbildung 3.2-24 ermittelt. Der Lotfußpunkt beschreibt den Schnittpunkt der Spiegelachse mit ihrer Normalen durch den kollidierenden Voxel.

$$P_{\text{Lotfußpunkt}} = P_1 + v ((P_{\text{kollidierender Voxel}} - P_1) \cdot v)$$

$$P_{\text{kollidierender Voxel,neu}} = P_{\text{kollidierender Voxel}} + 2(P_{\text{Lotfußpunkt}} - P_{\text{kollidierender Voxel}})$$

Formel 3-15 Spiegelung

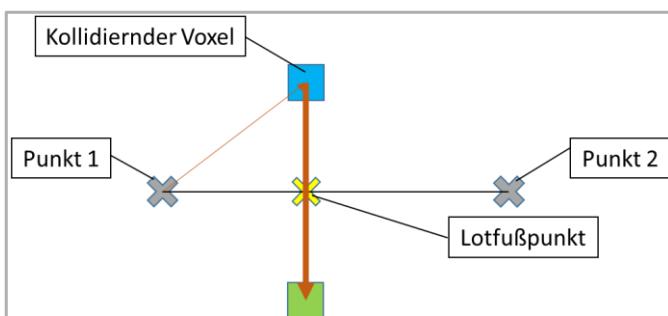


Abbildung 3.2-24 Skizze Spiegelverfahren

Abbildung 3.2-25 veranschaulicht die Spiegelung aller Kollidierenden Voxel (lila) an der Spiegelachse (Rote Markierung) an ihre neue Position (Gelb).

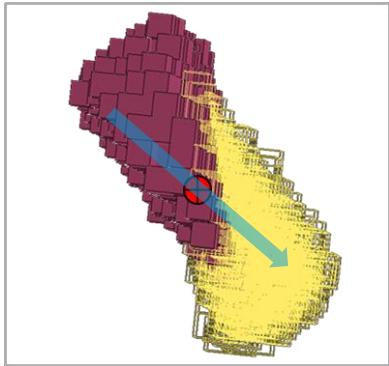


Abbildung 3.2-25 Spiegelung an der Geraden

Die Abbildung 3.2-26 stellt den Zustand des betrachteten Übergangs nach der Ausführung der Spiegelfunktion dar. Die gespiegelten Voxel sind gelb markiert. Zugeordnet werden diese dem roten Extrusionskörper. Die Positionen der Voxel des grünen Körpers wurden in diesem Beispiel nicht verändert.

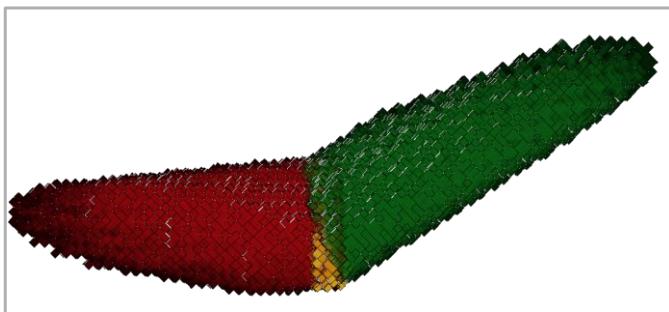


Abbildung 3.2-26 Zustand nach der Spiegelung. Die vormals kollidierenden Voxel sind in Gelb dargestellt

3.2.4 Kollisionsbedingtes Umformen

3.2.4.1.1 Funktionsaufbau

Nach der Beseitigung der rovinginternen Kollisionen im vorangegangenen Schritt werden innerhalb der im folgenden beschriebenen Funktion die Kollisionen verschiedener Rovinge durch eine lokale Veränderung des Rovingquerschnitts beseitigt. Hierdurch bleibt das Faservolumen der einzelnen Rovinge erhalten. Die Funktion ist im Vergleich mit dem bereits beschriebenen Verfahren ungleich komplexer, weshalb einige Erläuterungen sowohl in der allgemeinen Beschreibung des Funktionsaufbaus als auch in der genauen Beschreibung der entsprechenden Unterfunktion enthalten sind.

Im ersten Schritt werden im folgenden sogenannte Kollisionskörper ermittelt. Diese bestehen aus mehreren kollidierenden und zusammenhängenden Voxeln. Diese Kollisionskörper werden danach sequentiell abgearbeitet. Innerhalb der Kollisionskörper werden wiederum die vorkommenden Extrusionskörper gemäß Ihrer Shell-ID nacheinander bearbeitet (vgl. Abbildung 3.2-27).

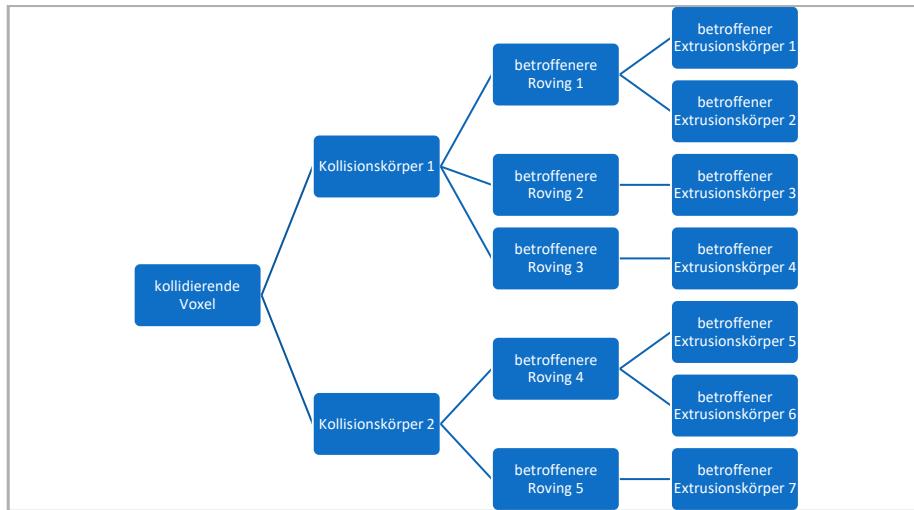


Abbildung 3.2-27 Gliederung und Zuordnung der kollidierenden Voxel

Wie in Abbildung 3.2-28 zu erkennen ist wird zunächst eine Schnittfläche durch die Randflächen der kollidierenden Rovinge bestimmt. An dieser werden die Voxel gespiegelt. Abschließend werden die Voxel dann parallel zur ermittelten Schnittfläche und Normal zur Faserrichtung auf noch unbesetzte Positionen im Voxelgerüst verschoben.

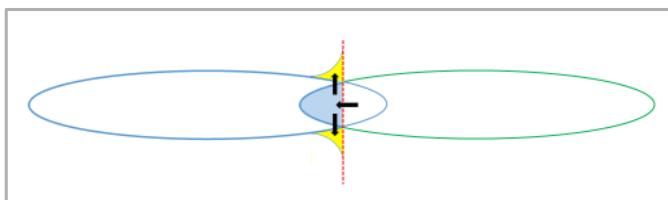


Abbildung 3.2-28 Vorgehensweise der Umformung

3.2.4.1.2 Validierung des Verformungsalgorithmus

Die implementierte Verformungsvorschrift bildet das experimentell ermittelte Verhalten der Rovinge bei gegenseitigem Kontakt nach. An den markierten Positionen der Abbildung 3.2-29 ist zu erkennen, dass sich der Rovingquerschnitt an die Kontaktfläche anschmiegt.

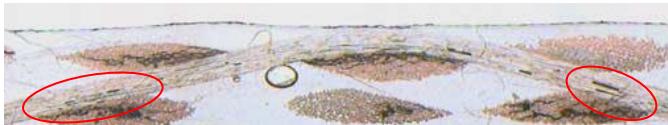


Abbildung 3.2-29 Faserverbund Querschnitte [33]

Bestätigt wird diese Beobachtung auch in Abbildung 3.2-30 und durch die Arbeit von Baoxing Chen und Tsu-Wei Chou [34], die den Rovingquerschnitt als teilweise elastischer Körper modellierten.

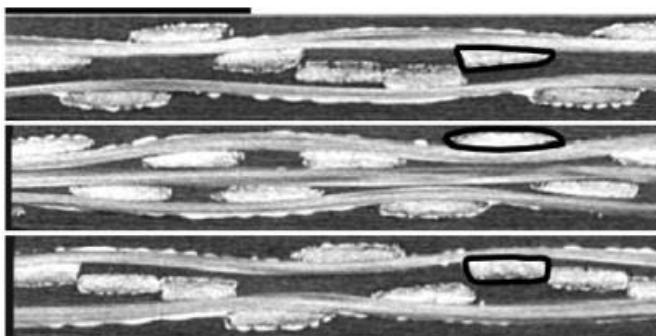


Abbildung 3.2-30 Faserverbund Querschnitte 2 (13)

3.2.4.1.3 Ermittlung der Kollisionskörper

Für die Bestimmung der Kollisionskörper werden wiederrum mittels der in Abschnitt 3.2.3.1.2 vorgestellten Vorgehensweise alle kollidierenden Voxel bestimmt. Für die so bestimmten Voxel werden die Positionsindexe der acht Knoten bestimmt, welche als Ecken der Würfel (siehe Abbildung 3.2-31) diese in dem Referenzgitter von Abschnitt 3.1.1.1.4 definieren.

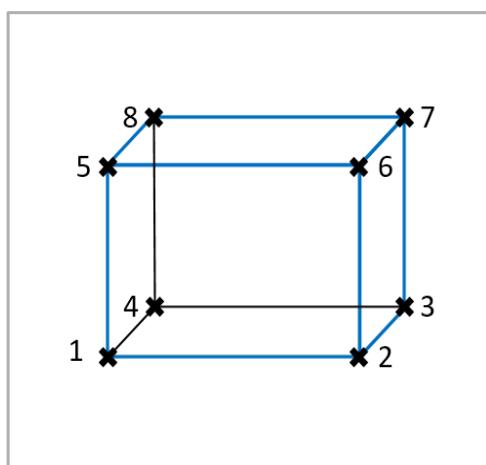


Abbildung 3.2-31 Aufbau eines Solids

Die kollidierenden Voxel tasten in zufälliger Reihenfolge nacheinander ihre umgebenden Gitterzellen nach Voxel ab, welche bereits einem Kollisionskörper zugewiesenen sind. Über gemeinsame Knoten-Indizes werden hierbei benachbarte Voxel gefunden und dem gleichen Kollisionskörper zugewiesen. Sollten durch das sich gerade in Bearbeitung befindliche Voxel mehrere bereits bestehende Kollisionskörper miteinander verknüpft werden, so werden deren gesamte Voxel dem minimal vorkommenden Kollisionskörper zugewiesen. Falls keine gemeinsamen Knoten mit einem bereits zugewiesenen Voxel gefunden werden, wird der Voxel einem neuen Kollisionskörperindex zugewiesen.

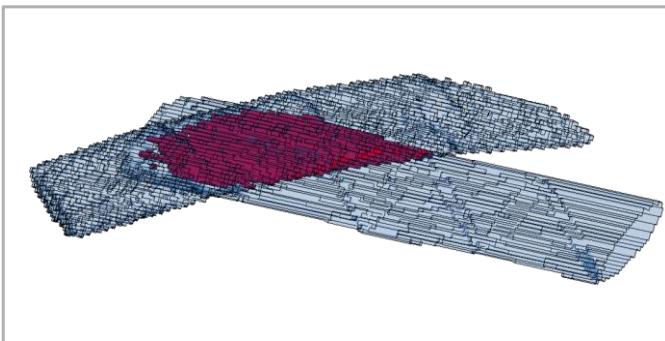


Abbildung 3.2-32 Ermittlung eines Kollisionskörpers

3.2.4.2 Verschiebung in Richtung der Rovingmitte

3.2.4.2.1 Ermittlung der Schnittfläche

Die Ermittlung der Schnittflächen erfolgt auf Rovingebene (siehe Abbildung 3.2-27). Zunächst werden alle Randvoxel bestimmt, die zum aktuellen Roving und den von dem Kollisionskörper betroffenen Extrusionskörper gehören. Die Bestimmung der Randvoxel erfolgt wie die Suche nach benachbarten Kollisionsvoxeln im vorhergehenden Abschnitt über die Ermittlung der Positionsindexe für die Eck-Knoten-Indizes der Voxel. Da ein solcher Index, sofern der Voxel auf allen Seiten von anderen Voxeln umgeben ist, in insgesamt acht verschiedenen Voxeln vorkommen muss (vgl. Formel 8.3-17), bedeutet dies im Umkehrschluss, dass ein weniger als acht Mal vorkommender Knotenindex am Rand der Struktur liegen muss. Aufgrund dieser Erkenntnis werden alle Voxel, welche mindestens einen solchen Randknoten verwenden als Randvoxel gekennzeichnet. Im nächsten Schritt werden die Minima und Maxima in X-, Y- und Z-Richtung aller Voxel, die zum aktuell betrachteten Kollisionskörper gehören und als Randvoxel markiert wurden, bestimmt. Für die Ermittlung der Schnittfläche werden nun diejenigen Koordinatenrichtungen mit den beiden größten Differenzen zwischen Maxima und Minima der vorkommenden Randvoxel ausgemacht. Als Punkt 1 für die Definition der Schnittfläche wird ein Voxel gewählt, der in Richtung der größten Differenz das Minimum aufweist. Als Punkt 2 wird ein Voxel mit dem maximalen Wert in Richtung der größten Differenz herangezogen. Der 3. Punkt wird durch das Voxel mit dem maximalen Abstand zur Geraden Punkt1-Punkt2 bestimmt. Die Berechnung des Abstands aller Randvoxel, die weder Punkt 1 noch Punkt 2 sind erfolgt indem diese in ein weiteres Koordinatensystem transformiert werden. Die Basisvektoren werden hierbei aus dem Einheitsvektor in Richtung der minimalen Differenz, der Geraden zwischen Punkt 1 und Punkt 2 und der Normalen zu der durch die beiden genannten Vektoren definierten Fläche gebildet. Als Ursprung des Koordinatensystems wird Punkt 1 angenommen. Der Abstand zur genannten Geraden ergibt sich aus dem absoluten Wert der Randvoxel in Richtung der definierten Normalen. Durch die drei Punkte wird die Schnittebene festgelegt. Aus der Schnittebene wird mittels eines Kreuzproduktes abermals ein Koordinatensystem relativ zum Ursprung in Punkt 1 definiert. Die Richtung des Normalenvektors wird ermittelt, indem die Anzahl aller nicht kollidierender Voxel des betrachteten Extrusionskörpers berechnet wird. Liegt die Mehrheit der so ermittelten Voxel unterhalb der Schnittebene, so wird die Richtung des bestimmten Normalenvektors umgekehrt.

3.2.4.2.2 Verschiebungsfunktion

Die kollidierenden Voxel werden zunächst in das vorher bestimmte, als „Schnittebenenfestes“ Koordinatensystem bezeichnete, Koordinatensystem umgerechnet. Durch die Verschiebungsfunktion werden nun alle Voxel mit negativem Vorzeichen (blaue Voxel) in der Normalenrichtung zur Schnittebene (grüne Linie) an sich selbst gespiegelt. Dies erfolgt durch Multiplizieren des entsprechenden Wertes mit -1 (gelbe Voxel).

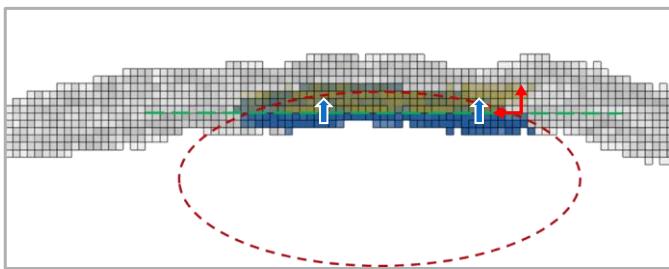


Abbildung 3.2-33 Skizze der Spiegelung an der Schnittebene

3.2.4.3 Verschiebung parallel zur Schnittfläche

3.2.4.3.1 Funktionsaufbau

Die Verschiebung der kollidierenden Voxel parallel zur Schnittfläche schließt die Umformfunktion ab. Im Gegensatz zur Bearbeitung der einzelnen Rovinge, bzw. aller von einem Kollisionskörper betroffenen Extrusionskörper, werden innerhalb dieser Funktion die Extrusionskörper (siehe Abbildung 3.2-27) einzeln bearbeitet. Dieses Vorgehen ist notwendig, um die Verschiebungsrichtung der Voxel normal zur Faserrichtung durchzuführen, welche für jede Shell, bzw. deren Extrusionskörper individuell festgelegt ist. Die Bearbeitung findet wiederum in einem eigens für die Funktion erstellten Koordinatensystem statt. Die Bestimmung der Richtungsvektoren geschieht nach dem folgenden Prozess:

1. Die Faserrichtung wird aus der Extrusionsfunktion übernommen.
2. Die Normale zur Schnittfläche wird vom aktuell bearbeiteten Kollisionskörper/Roving-Kombination übernommen
3. Durch die Berechnung des Kreuzproduktes wird der dritte, senkrecht zu den bereits bestehenden Vektoren berechnet.
4. Die Normale zur Schnittfläche wird durch das Kreuzprodukt aus dem Vektor in Faserrichtung und dem unter 3. berechneten Vektor ersetzt. Somit ergibt sich ein kartesisches Koordinatensystem.

Als Ursprung wird ein beliebiger Voxel referenziert. Die in der Funktion 3.2.4.2 an der Schnittfläche gespiegelten Voxel werden in das neue Koordinatensystem transformiert. Da keine Verschiebung in Faserrichtung möglich ist, findet die weitere Verarbeitung „Scheibenweise“ statt. Gebildet werden die Scheiben, wie in Abbildung 3.2-34 erkennbar ist, normal zur Faserrichtung. Die Bearbeitung der Scheiben erfolgt sequentiell.

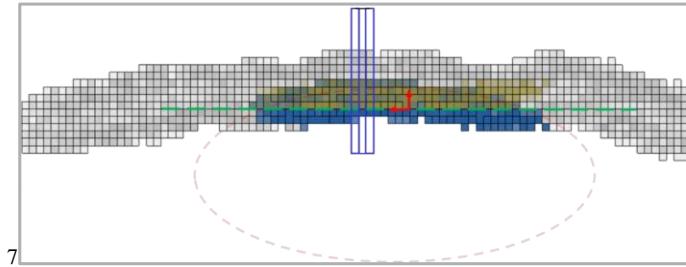


Abbildung 3.2-34 Skizze der Scheibenermittlung

Die in der jeweiligen Scheibe enthaltenen Voxel werden nun nach Möglichkeit horizontal zur Schnittebene auf freie, angrenzende Gitterpositionen im globalen Koordinatensystem verschoben. Sollten durch eine hohe Faserdichte nicht genügend freie Positionen im Gitter für die horizontale Verschiebung vorhanden sein, so werden die entsprechenden Voxel vertikal in Richtung des unter 4. bestimmten Vektors verschoben. Begrenzt werden die Voxel durch den maximalen Wert aller Voxel des aktuell betrachteten Extrusionskörpers in dieser Richtung. Schlussendlich werden die Voxel wieder zurück in das globale Koordinatensystem zurücktransformiert.

3.2.4.3.2 Freie Voxel bestimmen

Vor der horizontalen Verschiebung werden für alle vorkommenden Werte des Y-Vektors, welcher senkrecht auf der Schnittebene steht die Randvoxel ermittelt. Aus den Randvoxeln werden sogenannte „Potentielle Freie Voxel“ ermittelt, indem allen Randvoxel, wie in Abbildung 3.2-35 dargestellt, in horizontaler Richtung der Wert 1 bzw. -1 hinzugefügt wird.

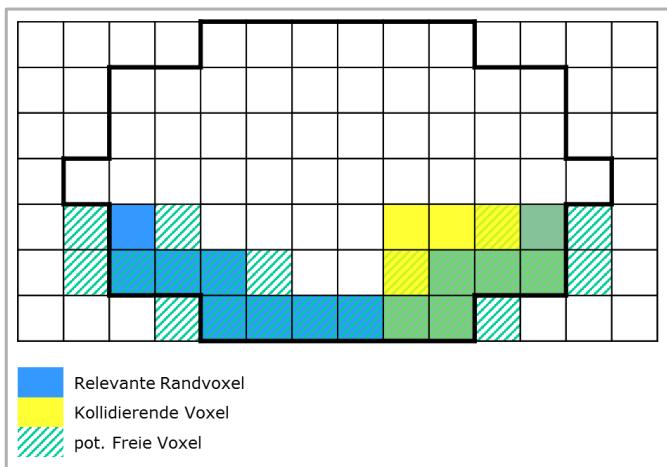
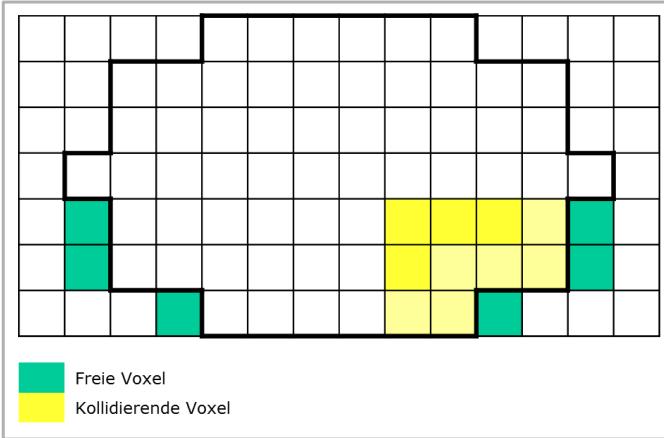


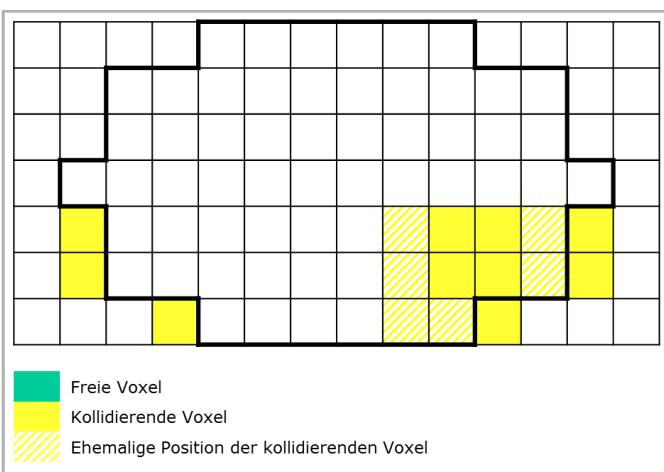
Abbildung 3.2-35 Ermittlung der pot. freien Positionen

Diese möglichen freien Voxel werden nun in das globale Koordinatensystem zurücktransformiert. Alle auf einer noch unbesetzten Gitterposition positionierten möglichen freien Voxel werden als wirkliche freie Voxel übernommen (Abbildung 3.2-36).



3.2.4.3.3 Horizontale Verschiebung

Die horizontale Verschiebung findet innerhalb der in Abbildung 3.2-34 definierten Scheiben wiederum ebenenweise statt. In jeder Ebene wird die maximal mögliche Anzahl an zu verschiebenden Voxeln auf die im vorangegangenen Abschnitt gefundenen freien Positionen verschoben. Dies ist in Abbildung 3.2-37 skizziert.



Die verschobenen Voxeln werden hierdurch zu Randvoxeln (vgl. Abbildung 3.2-38), sodass für die selbe Ebene erneut freie Voxel bestimmt werden können. Die Liste der freien Voxel wird um die verschobenen Voxel am Ende der Verschiebung verkürzt. Als nächster Schritt werden wieder gemäß Abbildung 3.2-35 potentielle freie Voxel ermittelt. Nachdem alle kollidierenden Voxel verschoben wurden, wird die nächste Scheibe bearbeitet.

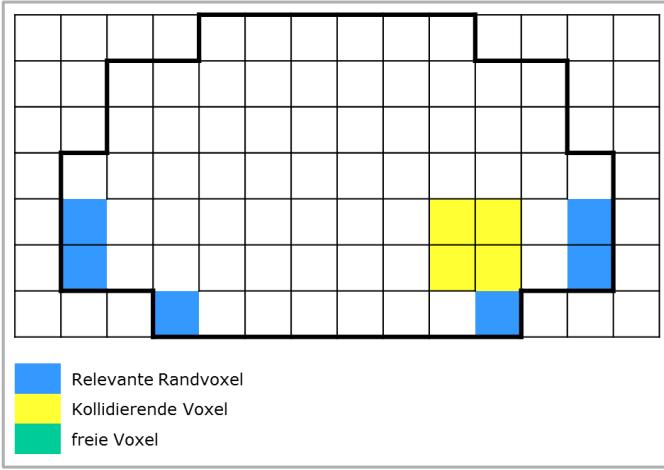


Abbildung 3.2-38 Skizze der neuen Randvoxel

3.2.4.3.4 Vertikale Verschiebung

Falls auf einer von einem zu verschiebenden Voxel besetzten Ebene, weder ein freier Voxel, noch ein Randvoxel und somit kein möglicher freier Gitterplatz verfügbar ist (siehe Abbildung 3.2-39), werden die betroffenen Voxel innerhalb der Scheibe nach oben verschoben (vgl. Abbildung 3.2-40). Die Verschiebung in dieser Richtung wird jedoch durch den maximal vorkommenden Wert aller Randvoxel der Scheibe begrenzt.

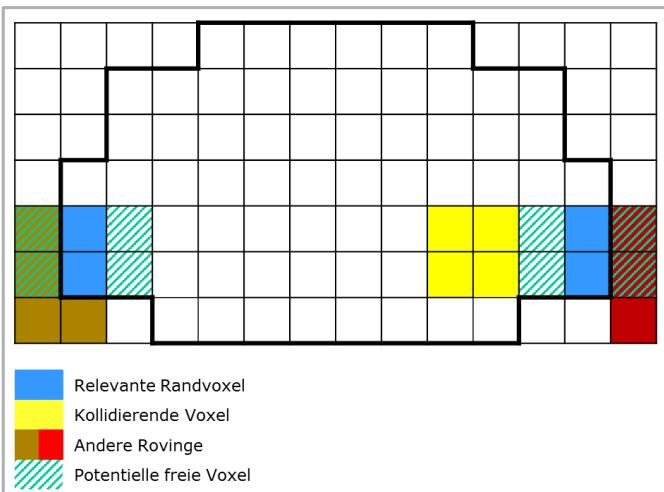


Abbildung 3.2-39 Problematik bei anliegenden fremden Rovingen

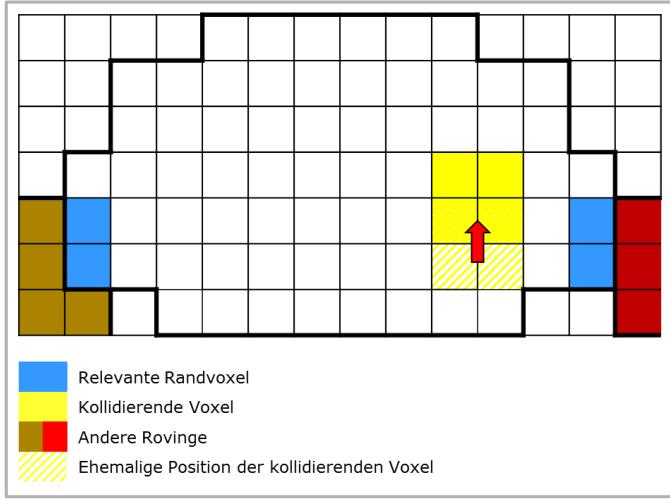


Abbildung 3.2-40 Vertikale Verschiebung

3.3 VERARBEITUNG DER KAVITÄTSDATEN

3.3.1.1 *Funktionsaufbau*

Ziel der vorliegenden Funktion ist die auf die erzeugten Faserdaten abgestimmte Beschreibung des Kavitätsraumes durch Voxel. Hierbei sollen im Sinne einer effizienten Programmierung möglichst viele bereits für die Faserdaten genutzten Funktionen und Methoden verwendet werden. Die Funktion besteht aus fünf Unterfunktionen, die einmalig hintereinander aufgerufen werden. Zunächst wird die vom Nutzer bestimmte Datei eingelesen und die enthaltenen Daten in von Matlab nutzbare Matrizen konvertiert. Die Eingabedaten beschreiben mittels drei- und viereckiger Shells die Randflächen der Kavität. Im zweiten Schritt wird analog zu 3.2.1.1.4 ein Referenzgitter für die Umgebung der Kavität definiert. In Schritt drei werden die Flächen in Voxel konvertiert, bevor im Schritt vier der so erzeugte noch hohle Körper mit Voxel gefüllt wird. Im fünften und letzten Schritt werden die Querschnitte für den Ein- und Ausfluss des Harzes festgelegt.

3.3.1.2 *Einlesen der Eingabedaten*

Die Funktion zum Einlesen der Kavitätseingabedaten ist weitestgehend identisch mit der der Faserdaten (vgl. 3.2.1.1.2). Der Aufbau der „.pc“-Datei ist analog zur Datei mit den Faserdaten. Im Gegensatz zu den initialen Faserdaten beschreiben die in der Datei enthaltenen Shells nicht den Verlauf der zu untersuchenden Rovinge, sondern beschreiben die Randflächen der Kavität. Wie in Abbildung 3.3-1 zu erkennen ist, kommen in der eingelesenen Datei eine Vielzahl an Vierecken neben einigen wenigen Dreiecken vor..

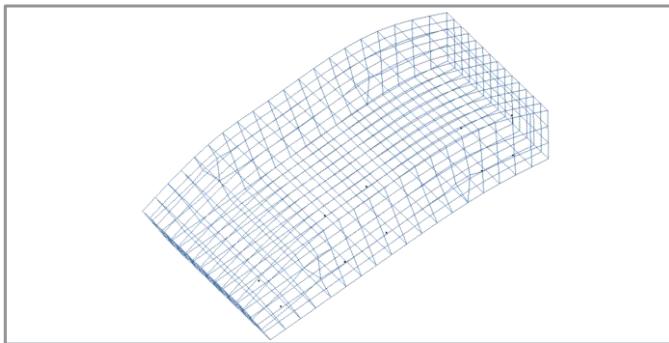


Abbildung 3.3-1 Visualisierung der Eingabedaten

Um die bereits bestehenden Funktionen, insbesondere die Extrusionsfunktion 3.2.2 nutzen zu können, werden die Vierecke in jeweils zwei Dreiecke konvertiert. Es entsteht Abbildung 3.3-2.

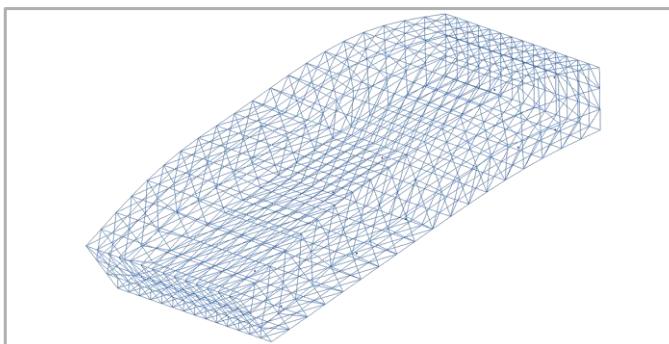


Abbildung 3.3-2 Visualisierung der in Dreiecke konvertierte Eingabedaten

3.3.1.1.3 Voxelerzeugung aus den Randflächen

Die Umwandlung der bestehenden Shells in Voxel erfolgt mit der bereits bekannten Funktion „extrudieren“ (Siehe 3.2.2). Im Gegensatz zu der im Rahmen der Faserdatenverarbeitung genutzten Funktion, wird nun die Querschnittshöhe auf eine Voxelkantenlänge festgelegt. Nach der Extrusion aller Randflächen ergibt sich die Abbildung 3.3-3 in der Gesamtansicht und die Abbildung 3.3-4 in der Schnittansicht. Die eingelesenen Randflächen werden jeweils in blau dargestellt.



Abbildung 3.3-3 Randflächen (blau) inkl. erzeugter Voxel

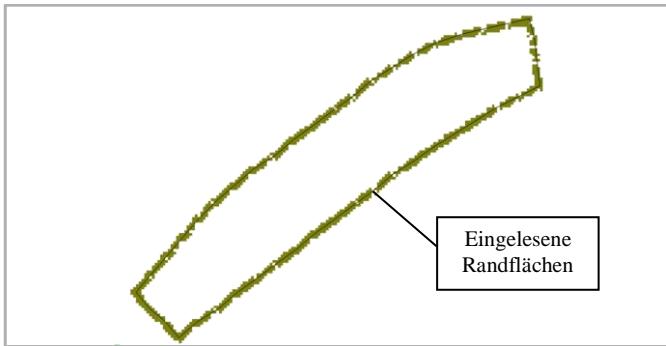


Abbildung 3.3-4 Querschnitt der Randflächen inkl. der erzeugten Voxel

3.3.1.1.4 Auffüllen der Kavität

Die leere Voxelhülle wird in die vom Nutzer vorgegebene Richtung aufgefüllt, indem gemäß Abbildung 3.3-5 zunächst alle Voxel der Hülle auf die zur Injektionsrichtung senkrechte Ebene projiziert werden. Die Fülfunktion ermittelt nun für alle vorkommenden Indizes nacheinander jeweils die in Füllrichtung vorkommenden Minima und Maxima der Voxelhülle. Als Füllung der Hülle resultieren hieraus alle Voxel, die für den jeweiligen ermittelten Index zwischen dem Minimum und Maximum liegen. In Abbildung 3.3-5 wird dieser Vorgang durch die blauen Balken veranschaulicht. Die nicht in Füllrichtung zeigenden Koordinaten werden durch die Projektion auf die Ebene bestimmt.

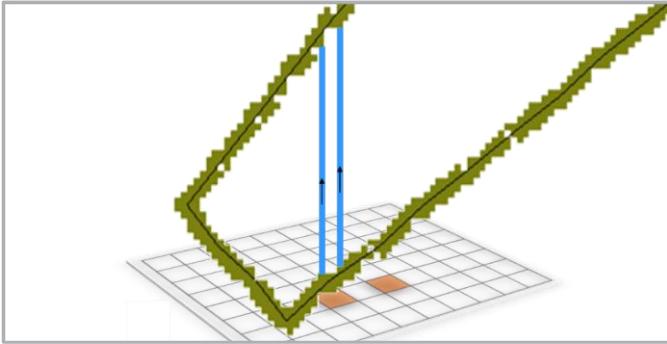


Abbildung 3.3-5 Befüllungsvorgang inkl. Projektionsfläche

Da es durch Rundungsfehler bei der Erstellung der Hülle zu Löchern in der Kavitätshülle kommen kann (siehe Markierung in Abbildung 3.3-6), wird aufgrund des fehlenden Minimums oder Maximums keine Befüllung für die jeweilige Stelle auf der Projektionsebene durchgeführt. Daher wird die Füllfunktion analog zur bisher beschriebenen Vorgehensweise für die beiden verbliebenen Richtungen durchgeführt.

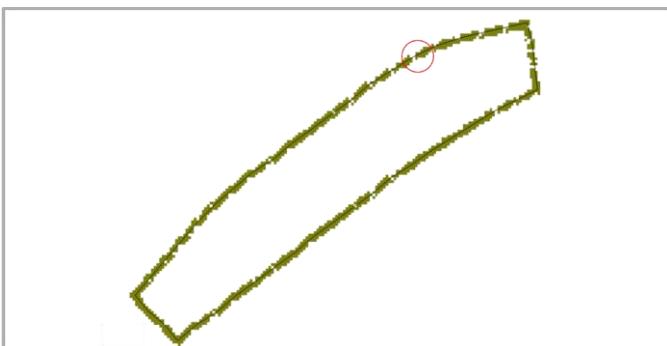


Abbildung 3.3-6 Darstellung möglicher "Löcher" in erzeugten Voxelhülle

Hierdurch kommen praktisch keine Hohlräume oder Tunnel in Füllrichtung in der Kavität mehr vor. Die Hülle wird somit dreifach gefüllt. Durch das Zusammenführen der Füllungen wird jedoch jede Gitterposition lediglich einfach genutzt.

3.3.1.1.5 Bestimmung der In- und Outlets

Die Bestimmung der In- und Outletvoxel erfolgt bei der Anwendung der Füllfunktion in Injektionsrichtung. Die ermittelten Minima werden als sogenannte „Inletvoxel“ durch in der Typspalte der Voxel gekennzeichnet, die Maxima analog als „Outletvoxel“. Es entsteht hieraus die Abbildung 3.3-7 mit farblich abgesetzten Voxeln.

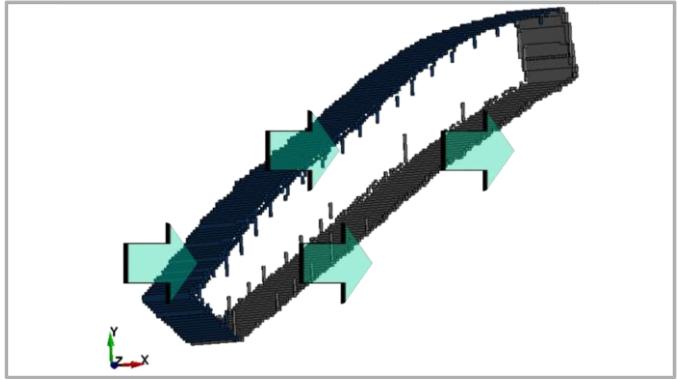


Abbildung 3.3-7 Darstellung der In-/Outletflächen inkl. Harzinjektionsrichtung

3.4 AUSGABEFUNKTION

3.4.1.1.1 Zusammenfügen der erzeugten Daten

Die aus den Funktionen 3.2 und 0 erzeugten Voxellisten werden zu Beginn der Ausgabefunktion zusammengeführt. Hierfür werden sowohl die Faser- als auch die Kavitätsvoxel in ein gemeinsames Koordinatensystem transponiert. Im nächsten Schritt werden alle Voxel der Kavität, die sich eine Gitterposition mit einem Faservolumenvoxel teilen, entfernt. Die Ermittlung der kollidierenden Voxel erfolgt über die Ermittlung der Indexe analog zum Abschnitt 3.2.3. Die Faservoxel verdrängen somit die Kavitätsvoxel. Alle verbleibenden Voxel der Kavität werden dem Roving, bzw. richtigerweise dem Part „1“ zugeordnet. Die Part-/Roving-IDs der Faservoxel werden entsprechend um 1 erhöht.

3.4.1.1.2 Generierung der Solids

Das Ausgabeformat des Programms „dyn“ verfügt über einen ähnlichen Aufbau wie das für die Eingabedateien verwendete „.pc“-Format. Die direkte Ausgabe und visuelle Darstellung der zuvor erzeugten Voxelliste ist somit nicht möglich. Deshalb werden die bestehenden Voxel wiederrum (vgl. 3.2.4.1.3) in Solids umgeformt, deren Form durch die Position im Raum der acht an den Ecken des Solids referenzierten Knoten definiert wird. Durch das strukturierte Gitter (siehe 3.1.1.1.4, 3.5.1.1.2) und die Würfelform der Voxel (3.1.1.1.2) lassen sich die jeweiligen Knoten für jeden Voxel schnell erzeugen. Die Bezeichnung/ID eines jeden Knoten entspricht der eigenen, durch die Matlabfunktion `sub2ind` aus den Knotenkoordinaten ermittelten Index. Die Abbildung 3.4-1 zeigt den Aufbau eines Solids. Die Bezeichnungen der Knoten in der Abbildung entsprechen jedoch nicht der tatsächlichen Knoten-ID sondern beschreibt die Knotennummer (Spalte in Tabelle 3.4-1 + 2).

Solid-ID	Part-ID	Knoten 1	Knoten 2	Knoten 3	Knoten 4	Knoten 5	Knoten 6	Knoten 7	Knoten 8
1	1	11	12	13	14	15	16	17	18
2	1	15	16	17	18	21	22	23	24

Tabelle 3.4-1 Solids Tabelle zweier aufeinanderliegender Voxel

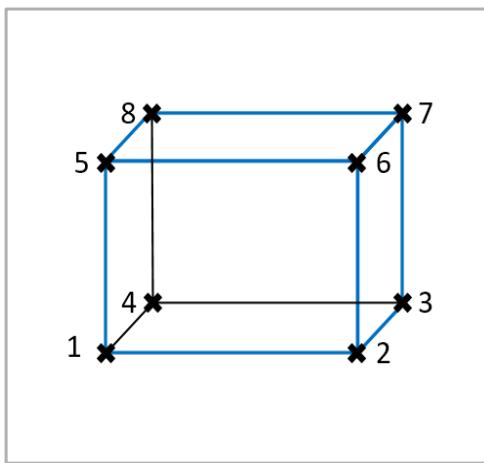


Abbildung 3.4-1 Aufbau eines Solids

3.4.1.1.3 Generierung der Knotendaten

Das Format der Knotenausgabe entspricht dem der Eingabedaten (siehe 3.2.1.1.2). Zunächst werden alle in der Solidstabelle (vgl. Tabelle 3.4-1) in den Spalten 3 bis 10 vorkommenden Knoten-IDs ermittelt. Da die ID jedes Knotens gleichzeitig seinem Positionsindex im Mesh entspricht werden die entsprechenden Koordinaten der Knoten mittels der Funktion `ind2sub` direkt berechnet. Die endgültigen

Koordinaten der Voxel werden mit der Formel XXX ermittelt. Die Relationen und Größen der Daten entsprechen somit wiederum denjenigen der anfangs eingelesenen Dateien.

$$\text{Koordinaten} = (\text{Koordinaten} + \text{Ursprung}) \cdot \text{Kantenlänge}$$

Formel 3-16 Rücktransformation der Koordinaten in das ursprüngliche Koordinatensystem

Knoten-ID	X-Pos. [Kantenlänge]	Y-Pos. [Kantenlänge]	Z-Pos. [Kantenlänge]
50	4	5	7
60	6	6	8

Tabelle 3.4-2 Knotenliste in Einheit Kantenlänge

Knoten-ID	X-Pos. [mm]	Y-Pos. [mm]	Z-Pos. [mm]
50	-0,6	0,5	-1,3
60	-0,4	0,6	-1,2

Tabelle 3.4-3 Knotenliste in Einheit mm

3.4.1.4 Generierung der Randflächen

Die Randflächen/Boundaries der einzelnen Rovinge respektive Parts werden bestimmt, indem zunächst alle aus den Randvoxeln (vgl. 3.2.4.2) erzeugten Solids in sechs voneinander unabhängige Flächen zerlegt werden. Der Aufbau der Flächen/Shells entspricht Kapitel 3.2.1.1.2 und wird in Tabelle 3.4-4 dargestellt.

Shell-ID	Part-ID	Knoten 1	Knoten 2	Knoten 3	Knoten 4
101	1	12	13	14	15
102	1	12	13	24	25

Tabelle 3.4-4 Randflächenliste

In Abbildung 3.4-2 werden die so entstandenen Randflächen in Rot dargestellt.

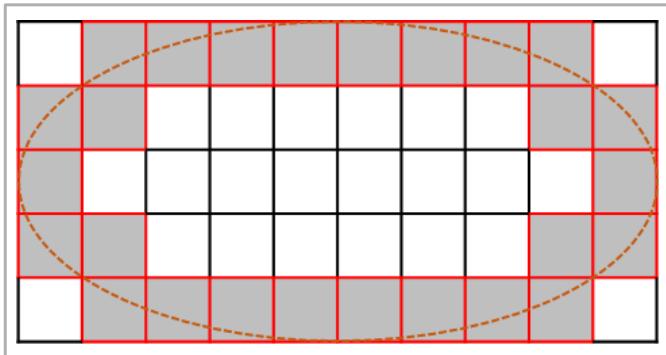
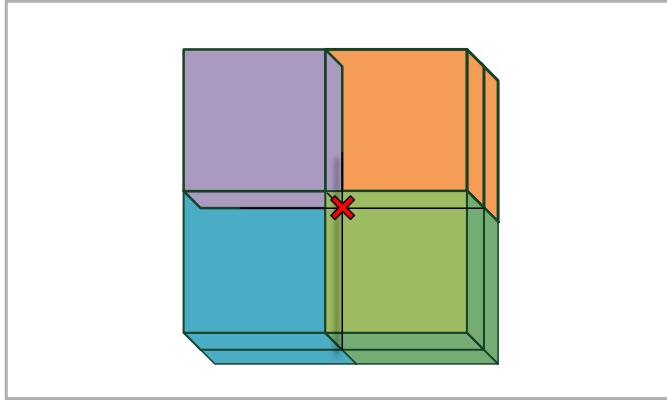


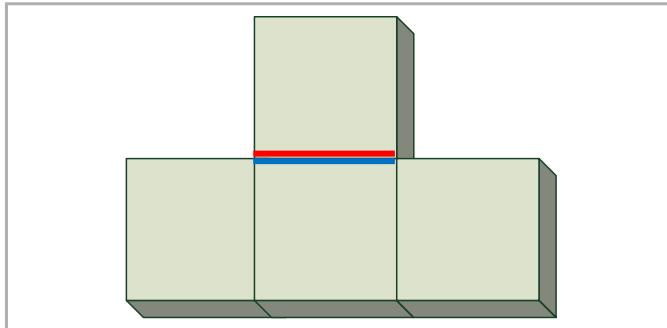
Abbildung 3.4-2 Darstellung der Randflächen vor der Filterung

Wie aus der Abbildung ersichtlich ist, handelt es sich nicht bei allen generierten Flächen um tatsächliche Randflächen. Weshalb im weiteren Verlauf die Menge der Flächen eingegrenzt wird. Hierfür wird die Anzahl der Referenzierungen im gesamten Roving bzw. der Kavität für alle in den Randsolids vorkommenden Knoten ermittelt. Wie in Formel 8.3-17 zu erkennen ist, wird der rot markierte Knoten von acht verschiedenen Solids genutzt. Bei allen Flächen, die den markierten Knoten verwenden handelt es sich um keine Außenfläche des dargestellten Körpers.



Formel 3-17 Veranschaulichung Knoten/Voxelzusammenhang

Aufgrund dieser Erkenntnis, werden alle Flächen, die über einen Knoten verfügen der in acht verschiedenen Solids verwendet wird gelöscht. In einem zweiten Filterschritt werden die in Formel 8.3-18 doppelt vorkommenden potentiellen Randflächen entfernt. Die in rot und blau hervorgehobene Fläche bleibt nach der zuvor beschriebenen Filterung erhalten obwohl es sich um keine Randfläche des dargestellten Körpers handelt. Die Menge der Randflächen wird also im Zuge der zweiten Filterung um alle mehrfach auftretenden Flächen verringert.



Formel 3-18 Veranschaulichung der Problematik bei hervorstehenden einzelnen Voxeln

3.4.1.1.5 Erzeugen der In- und Outletflächen

Alle Randflächen, die auf einem in Kapitel 3.3.1.1.5 definierten Inlet oder Outletvoxel basieren, werden entsprechend der Inlet oder Outletfläche zugewiesen, indem diese eine gesonderte Part-ID erhalten.

3.4.1.1.6 Einschränkungen

Augrund der Beschränkung der Knoten-ID auf 8 Stellen muss die Anzahl der referenzierten Knoten unter 100 Mio Stück liegen. Dies limitiert die Anzahl der erzeugten Voxel auf einen Wert zwischen 12.499.999 (vgl. Formel 3-19) und 99.252.847 (vgl. Formel 3-20). Im ersten Fall handelt es sich um einen Körper aus lediglich nicht zusammenhängenden Voxeln. Im zweiten Fall um einen Würfelförmigen Körper.

$$Anzahl_{minimal} = (1 \cdot 10^8 - 1) \div 8$$

Formel 3-19 Berechnung minimal mögliche Voxelanzahl

$$Kantenlänge_{Würfel}^3 + 2 \cdot Kantenlänge_{Würfel}^2 = 1 \cdot 10^8 - 1$$

$$Kantenlänge_{Würfel} = 463$$

$$Anzahl_{maximal} = Kantenlänge_{Würfel}^3$$

Formel 3-20 Berechnung maximale mögliche Voxelanzahl

3.5 OPTIMIERUNGSFUNKTIONEN

3.5.1.1.1 Gesamte Laufzeit

Die Laufzeit des Programms wird entscheidend von der durch den Nutzer in den Einstellungen (siehe 3.1.1.1.5) eingegebenen Daten beeinflusst. In Diagramm 3.5-1 wird die Entwicklung der Laufzeit über der Auflösung aufgetragen und sichtbar. Die exponentielle Laufzeitverlängerung durch die Verkleinerung der Kantenlänge ist direkt zu erkennen.

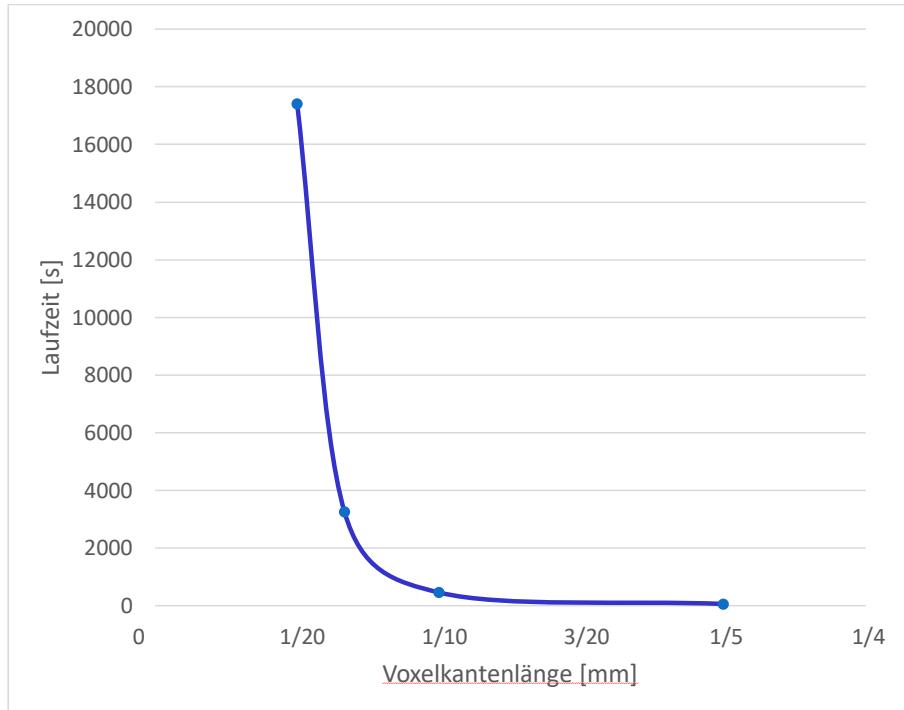


Diagramm 3.5-1 Laufzeit-Auflösungsdiagramm

Die Tests wurden direkt hintereinander auf dem unter 1 genannten System durchgeführt.

Die relative Laufzeit der einzelnen Programmodule hängt entscheidend von der Anzahl der kollidierenden Voxel und somit von der vom User eingestellt Rovinghöhe ab (vgl. 3.1.1.1.5. und 3.2.4). Die in Diagramm 3.5-2 wurde für die Rovinghöhe 0,5mm und die Auflösung 1/15mm ermittelt. Es ist eindeutig zu erkennen, dass der Großteil der Laufzeit auf die Auffüllfunktion der Kavität verwendet wird. Auffällig ist außerdem, dass die Extrusion der Shells (3.2.2) und die Herstellung der Verbindungen (3.2.3) in der Laufzeitbetrachtung nicht relevant ist.

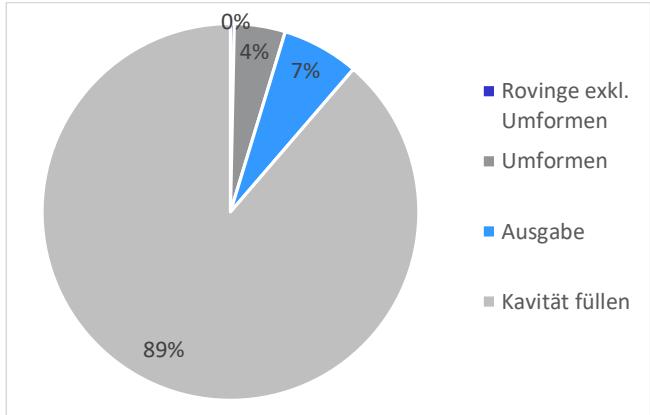


Diagramm 3.5-2 Relative Laufzeiten

3.5.1.1.2 Nutzen des geordneten Gitters

Im Folgenden wird die durch die Nutzung der Eigenschaften des geordneten Gitters in der Ausgabefunktion 3.4 erreichte Laufzeitverbesserung im Vergleich zu einem in einer früheren Entwicklungsversion genutzten Vergleichsprozess dargestellt. Im Gegensatz zu der unter 3.4 beschriebenen Funktion werden jedem Solid zunächst acht individuelle Knoten-IDs zugewiesen. Die Koordinaten der Knoten werden aus den Koordinaten des zugrundeliegenden Voxel abgeleitet. Im nächsten Schritt werden alle Knoten auf ihre einmalige Position kontrolliert und mehrfach vorkommende Knoten zusammengefasst. Abschließend werden die betroffenen Knoten-IDs in der Solids-Liste (siehe 3.4.1.1.2) durch die gemeinsame Knoten-ID ersetzt. Gemäß Diagramm 3.5-3 verkürzt sich die Laufzeit durch die genutzte Funktion um 96,6%.

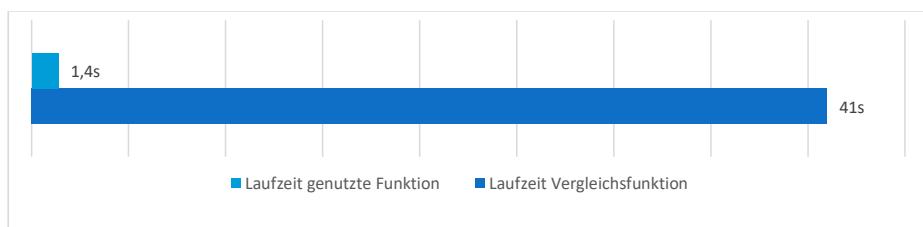


Diagramm 3.5-3 Laufzeitverbesserung durch geordnetes Gitter

Als Referenztabelle wurde eine Liste mit 10.000 zufällig platzierten Voxeln innerhalb eines Gitters mit $1 \cdot 10^6$ möglichen Positionen gewählt.

3.5.1.1.3 Indexierung

Bei der Suche oder dem Vergleich von Voxeln mit identischen Positionen lohnt es sich in Bezug auf die Laufzeit zunächst aus den drei Richtungskoordinaten einen eindeutigen Index für alle Voxel zu bilden und nach lediglich diese eindeutigen Indexwerte zu vergleichen anstatt direkt die einzelnen Koordinaten miteinander zu vergleichen. Dies macht sich besonders in den Funktionen 3.2.4 und 3.3.1.1.4 bemerkbar, da in diesen Funktionen in großem Ausmaß bereits nach bestehenden Voxeln gesucht wird. Belegt wird die These durch das Diagramm 3.5-4. Im Gegensatz zur schlussendlich genutzten Version sucht die Vergleichsversion direkt nach den zu ermittelnden Voxelkoordinaten, anstatt zunächst die bestehenden Voxel gemäß ihrer jeweiligen Position zu indexieren und daraufhin lediglich die gesuchten Indexwerte zu ermitteln. Für den Vergleich der beiden Funktionen wurden 10.000 Voxel aus einer Menge von 100.000 Voxel gesucht.

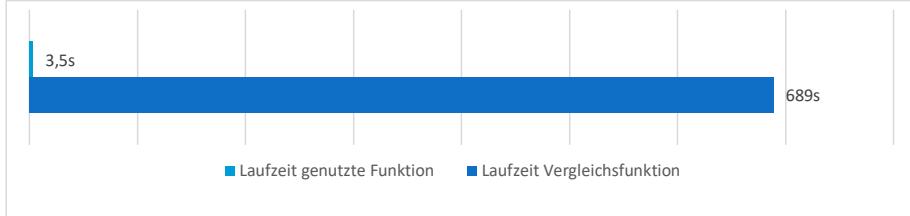


Diagramm 3.5-4 Vergleich Suchfunktion mit und ohne vorherige Indexierung

3.5.1.1.4 Nutzen von Listen

Wie in Absatz 3.2.2.1.1 beschrieben, findet die Verarbeitung der Faserdaten, mit Ausnahme der Extrusion (siehe 3.2.2.1.4) der einzelnen Shells, in der unter 3.1.1.1.3 eingeführten Liste statt. Alternativ wäre auch eine Abbildung aller Voxel analog zu der in der Extrusion genutzten Methode möglich. Benötigt würde eine Matrix mit den Dimensionen des unter 3.2.1.1.4 ermittelten Referenzgitters. Die geordnete Gitterstruktur wäre hierdurch auf die Matrix übertragen. Jedoch verhinderte die geringe Voxeldichte innerhalb des beschriebenen Raumes eine sinnvolle Anwendung, da auch für nicht belegte Gitterpositionen Arbeitsspeicher von der Funktion belegt wird. In Abbildung 3.5-1 werden die Arbeitsspeicherauslastungen des Gesamtmatrixkonzepts (links) und des verwendeten Listenkonzepts (rechts) verglichen. Es ergibt sich eine um ca. 7% geringere Auslastung bei der Wahl des Listenkonzepts bei einer Voxelkantenlänge von 1/15 mm.

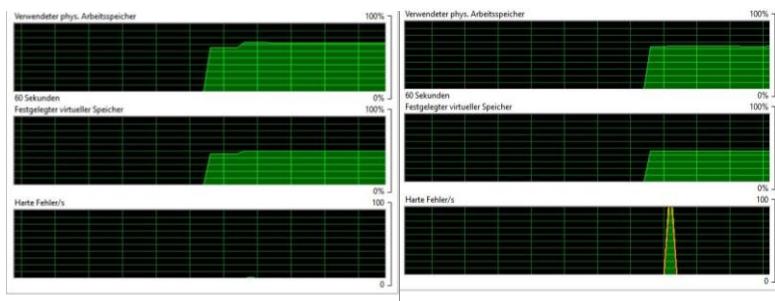


Abbildung 3.5-1 Vergleich Arbeitsspeicherauslastung Matrix-Liste

3.5.1.1.5 Verknüpfung der Voxel-IDs mit der Tabellenzeile

In den Funktionen 3.2.3 Erstellung der Übergänge und 3.2.4 Kollisionsbedingtes Umformen wird häufig lediglich eine Teilmenge aller Voxel bearbeitet. Um weitestgehend redundante Filtern und Suchen von bspw. allen kollidierenden Voxeln aus der Voxelliste zu vermeiden, werden diese aus der gesamten Liste herauskopiert. Um das Zusammenfügen der mittlerweile bearbeiteten Daten mit der Gesamtliste nach durchlauf der jeweiligen Unterfunktion effizient zu ermöglichen entspricht die Voxel-ID in der ersten Spalte der Voxelliste immer der Zeile des Voxels in der Gesamtliste. Hierdurch ergibt sich ein direkter Überschreiben-Befehl aus den bereits vorliegenden Daten. Durch den Verzicht auf eine Suchfunktion wird die Programmalaufzeit verkürzt. Wie in Diagramm 3.5-1 ersichtlich ist, spart dieses Vorgehen im Vergleich mit der alternativen Suchfunktion ca. 98,7% Laufzeit für den bereits in Kapitel 3.5.1.1.3 genutzten Testfall.

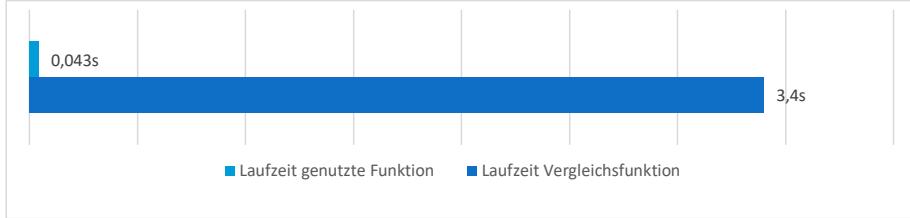


Diagramm 3.5-5 Laufzeitvergleich Zuordnung über ID und Zuordnung über Suchfunktion

3.5.1.1.6 Parallelisierung durch bsxfun-Funktion

Um die durch moderne Mehrkernprozessoren bestehenden Möglichkeiten der parallelisierten Programmausführung zu nutzen wird für die Anwendung von auf eine Vielzahl an ausführbare Befehle zurückgegriffen die eine parallele Bearbeitung von Knoten- oder Voxel-elementen ermöglichen. Im folgenden Beispiel sollen die 100.000 Voxel in der bereits vorgestellten Referenzliste entlang eines vorgegebenen Vektors verschoben werden. Durch das Programm genutzt wird die Funktion ***bsxfun***. Als Vergleichsfunktion dient eine ***for-Schleife*** die in einer frühen Programmversion genutzt wurde. Wie in Diagramm 3.5-6 ersichtlich wird reduziert sich die Rechenzeit um 99%.

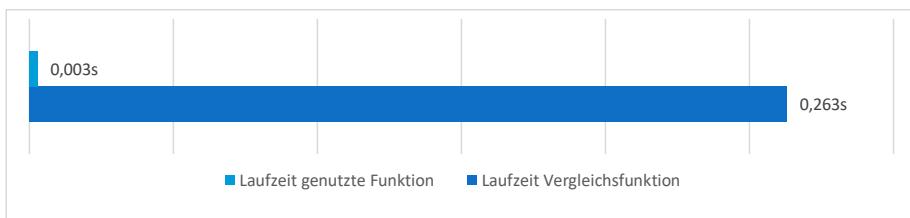


Diagramm 3.5-6 Laufzeitvergleich bsxfun vs. for-Schleife

4 Zusammenfassung und Ausblick

Die im Rahmen der Bachelorarbeit erarbeitete Methode der voxelbasierten Generierung von textilen Halbzeugmodellen bietet die Möglichkeit auch komplexe mesoskopische Volumen und Körper entkennbar abzubilden. Im Vergleich mit bestehenden Verfahren liegen die Vorteile der Methode im geringeren Bedarf an Rechenleistung und Zeit. Dies wird durch die Nutzung eines gleichförmig geordneten Gitters und eines reduzierten Voxelmodells ermöglicht. Hierdurch kann die Anzahl an genutzten Ermittlungs- und Suchfunktionen minimiert werden. Stattdessen greift das Programm wo immer möglich auf explizite Programmvorrichtungen zurück oder ordnet bestehende Elemente für die nachfolgenden Funktionen direkt passend an. Des Weiteren wird zur kollisionsbedingten Verformung statt relativ rechenintensiven Algorithmen, die auf der numerischen Lösung von Differentialgleichungen basieren, eine auf Experimenten basierende explizite Abbildungsvorschrift genutzt. Die stetige Optimierung der einzelnen Funktionen während der iterativen Programmierung machen sich neben der schlussendlichen Codeübersichtlichkeit besonders in der benötigten Laufzeit positiv bemerkbar. Neben der bereits erwähnten Substitution von impliziten Funktionen durch die Wahl geeigneter Methoden, Indexe oder Sortierungen, wurde die Laufzeit durch die Optimierung der noch vorhandenen Suchfunktionen bedeutend verkürzt. In diesem Zusammenhang sind besonders die Verwendung parallel verarbeitbarer Befehle der Matlabfunktionsbibliothek, sowie die Komprimierung mehrdimensionaler Daten auf einen einzelnen Indexwert, zu nennen. Durch die vorgestellten Umformmethoden zur Herstellung von Übergangen bzw. zur Vermeidung von Kollisionen bleibt die Rovingquerschnittsfläche und das Faservolumen über den gesamten Verlauf erhalten. Aufgrund der durch den Nutzer anpassbaren Einstellungen und der weitestgehend freien Wahl der Eingabefaserverläufe, lässt sich das erstellte Programm allgemein nutzen und bedarfsgerecht anpassen. Die Erweiterung des FasermODELLs durch die es umgebende Kavität erfüllt die in der Aufgabenstellung genannte Anforderung an die Weiterverwendbarkeit der erzeugten Geometrien durch CFD-Simulationsprogramme. Das Testen der Programmstabilität und eventuelle Anpassen des Programmes auf größere Faserstrukturen, als die der vorgestellten Einheitszelle, kann mit dem vorliegenden Tool erfolgen. Des Weiteren lässt sich mit den ausgegebenen Daten im Rahmen einer Versuchsreihe die maximal mögliche Kantenlänge für die nachfolgende CFD-Simulation ermitteln. Durch das Zerteilen komplexer Einzelfunktionen in jeweils übersichtliche und eindeutig definierte Unterfunktionen, ist die Wart- und Erweiterbarkeit des erstellten Tools sichergestellt. Unterstützt wird dieses Ziel durch die offensichtliche Ähnlichkeit der genutzten Strukturen mit alltäglichen Dingen wie Spielwürfeln oder Legosteinen. Dies ermöglicht in Zusammenspiel mit der sich nah an der Programmstruktur befindlichen vorliegenden Bachelorarbeit, das einfache Begreifen der genutzten Methoden und Funktionen.

Hierdurch ist es auch zukünftig möglich das Programm im Rahmen weiter zu optimieren bzw. um neue Funktionen zu erweitern. In Bezug auf die Programmalaufzeit wäre die Konvertierung in die Programmiersprache C++ oder C, neben der Nutzung von stark parallelisierten Rechnern wie beispielsweise CUDA eine vielversprechende Optimierungsmöglichkeit. Auch die im Abschnitt 3.5.1.1.4 verworfene Methode der Nutzung einer Abbildung des gesamten genutzten Raumes in einer dreidimensionalen Matrix, könnte bei entsprechenden Arbeitsspeicherreserven die Programmalaufzeit weiter verkürzen. Hierfür muss der Programmcode bei Beibehaltung der beschriebenen Methodik jedoch stark angepasst werden. Durch den großen Anteil der durch die Füllfunktion benötigten Laufzeit an der Gesamtaufzeit, ist durch eine Optimierung dieser Funktion bzw. Anpassung der zugrundeliegenden Methode, eine substantielle Verbesserung in Bezug auf die Laufzeit zu erwarten.

LITERATURVERZEICHNIS

1. **D.Veit.** *Simulation in textile technology*. Philadelphia : Woodhead Publishing Limited, 2012. 978-0-85709-029-4.
2. **S.V. Lomova, 1, , , G. Huysmansa, Y. Luoa, R.S. Parnasa, 2, A. Prodromoua, I. Verpoesta, F.R. Phelanb.** Textile composites: modelling strategies. *Composites Part A: Applied Science and Manufacturing*. 2001, Bd. 32, 10.
3. **Stepan V. Lomova, , , Dmitry S. Ivanova, Ignaas Verpoesta, Masaru Zakob, Tetsusei Kurashikib, Hiroaki Nakaib, Satoru Hirosawab.** Meso-FE modelling of textile composites: Road map, data flow and algorithms. *Composites Science and Technology*. 2007, Bd. 9, 67.
4. **Sherburn, Martin.** Geometric and Mechanical Modelling of Textiles. *University of Nottingham repository*. [Online] 2007. [Zitat vom: 29. 09 2016.] <http://eprints.nottingham.ac.uk/10303/1/thesis-final.pdf>.
5. **Q.T. Nguyen, E. Vidal-Salléa, P. Boissea, , , C.H. Parkb, A. Saouabb, J. Bréardb, G. Hivetc.** Mesoscopic scale analyses of textile composite reinforcement compaction. *Composites: Part B*. 2013, Bd. 44, 1.
6. **Cawkell, A.E.** *Encyclopaedic Dictionary of Information Technology and Systems*. London : Bowker-Saur, 1993. 1857390369.
7. **S.R. Chidamber, C.F. Kemerer.** A metrics suite for object oriented design. *IEEE Transactions on Software Engineering* . 1994, Bd. 20, 6.
8. **V. Menon, A.E. Trefethen.** MultiMATLAB Integrating MATLAB with High Performance Parallel Computing. *Proceedings of the ACM/IEEE SC97 Conference (SC'97)* . 1997.
9. **S. V. Lomov, T. Mikolanda , M. Kosek & I. Verpoest.** Model of internal geometry of textile fabrics: Data. *The Journal of The Textile Institute*. 2007, Bd. 98, 1.
10. **Birkefeld, Karin.** *Virtuelle Optimierung von Geflecht-Preforms*. Stuttgart : Institut für Flugzeugbau, 2013.
11. **P. Potluri, I. Parlak, R. Ramgulam, T.V. Sagar.** Analysis of tow deformations in textile preforms subjected. *Composites Science and Technology*. 2006, Bd. 66.
12. **Baoxing Chen, Tsu-Wei Chou.** Compaction of woven-fabric preforms in liquid composite molding processes: single-layer deformation. *Composites Science and Technology*. 1999, Bd. 59.
13. **P. Badela, E. Vidal-Salléa, E. Maireb, P. Boissea,.** Simulation and tomography analysis of textile composite reinforcement deformation at the mesoscopic scale. *Composites Science and Technology*. 2008, Bd. 68, 12.
14. **Cherif, Chokri.** *Textile Werkstoffe für den Leichtbau*. Dresden : Springer-Verlag Berlin Heidelberg, 2011. S. 113. ISBN: 978-3-642-17991-4.
15. **Siebenpfeiffer, Wolfgang.** *Leichtbau-Technologien im Automobilbau*. Stuttgart : Springer Vieweg, 2014. 978-3-658-04024-6.
16. **Ignaas Verpoest, Stepan V. Lomov.** Virtual textile composites software WiseTex: Integration with micro-mechanical, permeability and structural analysis. *Composites Science and Technology*. 2005, 65.
17. **Stepan V. Lomov, Enrique Bernal , Dmitry S. Ivanov , Sergey V. Kondratiev, Ignaas Verpoest.** Homogenisation of a sheared unit cell of textile composites. *Revue Européenne des Éléments* . 2005, Bd. 14, 6-7.
18. **S. De Jong, R. Postle.** A General Energy Analysis of Fabric Mechanics Using. *Textile Research Journal*. 1978, Bd. 48, 3.

19. **Tianyong Zheng, Jing Wei, Zhengtao Shi, Tingting Li, Zhen Wu.** An Overview of Modeling Yarn's 3D Geometric Configuration. *Journal of Textile Science and Technology*. 2015, Bd. 1.
20. *Finite element simulation of textile materials at mesoscopic scale.* **Durville, Damien.** St. Petersburg : HAL, 2007. HAL Id: hal-00274046.
21. **A. Doitrand Ȑ, C. Fagiano, F.-X. Irisarri, M. Hirsekorn.** Comparison between voxel and consistent meso-scale models of woven. *Composites: Part A*. 73, 2015.
22. **Hyung Joo Kim, Colby C. Swan.** Voxel-based meshing and unit-cell analysis of textile composites. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*. 2003, Bd. 56.

ANHANG

1 Messprotokolle

Kantenlänge: 1/5mm

Profile Summary

Generated 14-Sep-2016 07:39:11 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
F00_Main	1	59.515 s	0.008 s	
F01_Einstellungen	1	31.451 s	0.001 s	
uitools/private/uigetputfile_helper	3	31.445 s	0.013 s	
FileChooser.FileChooser>FileChooser.show	3	31.139 s	0.002 s	
...gt:FileChooser.showPeerAndBlockMATLAB	3	31.039 s	0.003 s	
uiputfile	1	16.760 s	0.001 s	
...ooser>FileSaveChooser.doShowDialog	1	16.658 s	16.088 s	
uigetfile	2	14.690 s	0.002 s	
...ooser>FileOpenChooser.doShowDialog	2	14.285 s	13.010 s	
F03_0_Kavitaet	1	12.512 s	0.002 s	
HF_13_Solids_bestimmen	21	10.397 s	2.084 s	
F04_0_Ausgabe	1	9.855 s	0.004 s	
sub2ind	1105611	8.422 s	8.422 s	
HF_06_0_Auffuellen	1567	7.621 s	0.153 s	
HF_06_2_Liste_befuellen	1567	7.276 s	4.308 s	
HF_12_Randvoxel_bestimmen	3	7.117 s	0.012 s	
F02_0_Rovinge	1	5.689 s	0.003 s	
F03_2_0_Kavitaet_fuellen	1	5.537 s	0.008 s	
F02_3_0_Umformen	1	4.830 s	0.001 s	
HF_04_0_vorkommende_Zeilen	66470	4.004 s	1.277 s	
HF_02_0_Extrudieren	2	3.835 s	0.202 s	
F03_3_0_InOutlets	1	3.636 s	0.532 s	
F04_5_Ausgabe_Datei	1	3.533 s	3.533 s	
F04_2_0_Shells_generieren	1	3.233 s	0.004 s	
F04_1_0_Solids_generieren	1	3.069 s	0.003 s	
ismember	66762	2.743 s	0.780 s	
F02_3_3_0_Umformen2	1	2.719 s	0.023 s	
F02_3_3_1_0_Verschiebungsfunktion	99	2.690 s	0.074 s	

Kantenlänge: 1/10mm

Profile Summary

Generated 14-Sep-2016 04:08:01 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
F00_Main	1	455.489 s	0.015 s	
F03_0_Kavitaet	1	335.302 s	0.005 s	
HF_06_0_Auffuellen	1567	301.376 s	0.186 s	
HF_06_2_Liste_befuellen	1567	300.975 s	177.981 s	
F03_2_0_Kavitaet_fuellen	1	296.211 s	0.056 s	
HF_04_0_vorkommende_Zeilen	188282	127.374 s	79.902 s	
HF_13_Solids_bestimmen	21	67.759 s	13.712 s	
F04_0_Ausgabe	1	62.813 s	0.014 s	
sub2ind	7351162	54.275 s	54.275 s	
ismember	188574	47.489 s	16.058 s	
HF_12_Randvoxel_bestimmen	3	46.788 s	0.053 s	
F01_Einstellungen	1	35.441 s	0.001 s	
uitools/private/uigetputfile_helper	3	35.437 s	0.012 s	
FileChooser.FileChooser>FileChooser.show	3	35.332 s	0.002 s	
...gt_FileChooser.showPeerAndBlockMATLAB	3	35.185 s	0.003 s	
F03_3_0_InOutlets	1	32.039 s	11.832 s	
ismember>ismemberR2012a	188574	31.431 s	28.442 s	
F02_0_Rovinge	1	21.919 s	0.002 s	
F04_2_0_Shells_generieren	1	21.299 s	0.030 s	
F04_5_Ausgabe_Datei	1	20.971 s	20.971 s	
F04_1_0_Solids_generieren	1	20.423 s	0.017 s	
uiputfile	1	19.102 s	0.000 s	
F02_3_0_Umformen	1	19.101 s	0.001 s	
...ooser>FileSaveChooser.doShowDialog	1	19.044 s	18.582 s	
uigetfile	2	16.338 s	0.002 s	
...ooser>FileOpenChooser.doShowDialog	2	16.069 s	15.108 s	

Kantenlänge: 1/15mm

Profile Summary

Generated 14-Sep-2016 03:57:13 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
F00_Main	1	3245.491 s	0.023 s	
F03_0_Kavitaet	1	2869.060 s	0.021 s	
HF_06_0_Auffuellen	1567	2712.929 s	0.436 s	
HF_06_2_Liste_befuellen	1567	2712.117 s	1528.901 s	
F03_2_0_Kavitaet_fuellen	1	2696.296 s	0.232 s	
HF_04_0_vorkommende_Zeilen	447706	1237.342 s	724.903 s	
ismember	447998	512.456 s	217.803 s	
ismember>ismemberR2012a	447998	294.652 s	249.577 s	
HF_13_Solids_bestimmen	21	228.380 s	44.655 s	
F04_0_Ausgabe	1	200.807 s	0.053 s	
sub2ind	24095124	184.443 s	184.443 s	
HF_12_Randvoxel_bestimmen	3	155.689 s	0.198 s	
F03_3_0_InOutlets	1	152.192 s	85.497 s	
F02_0_Rovinge	1	143.511 s	0.012 s	
F02_3_0_Umformen	1	133.738 s	0.004 s	
F02_3_3_0_Umformen2	1	78.203 s	0.116 s	
F02_3_3_1_0_Verschiebungsfunktion	141	78.075 s	1.528 s	
...3_1_2_0_Verschiebungsfunktion_Scheibe	2674	76.433 s	5.212 s	
F04_1_0_Solids_generieren	1	67.220 s	0.064 s	
F04_2_0_Shells_generieren	1	67.059 s	0.104 s	
F04_5_Ausgabe_Datei	1	66.112 s	66.112 s	
F02_3_3_1_2_1_freie_Voxel_funktion	10131	63.273 s	17.117 s	
ismember>ismemberBuiltInTypes	148672	45.075 s	45.075 s	
F01_Einstellungen	1	32.695 s	0.001 s	
uitools/private/uigetputfile_helper	3	32.692 s	0.011 s	
FileChooser.FileChooser>FileChooser.show	3	32.588 s	0.002 s	

Profile Summary

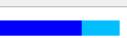
Generated 14-Sep-2016 12:39:02 using performance time.

<u>Function Name</u>	<u>Calls</u>	<u>Total Time</u>	<u>Self Time*</u>	Total Time Plot (dark band = self time)
F00_Main	1	17410.978 s	0.003 s	
F03_0_Kavitaet	1	16592.432 s	0.025 s	
HF_06_0_Auffuellen	1567	16117.276 s	0.427 s	
HF_06_2_Liste_befuellen	1567	16116.493 s	11707.766 s	
F03_2_0_Kavitaet_fuellen	1	16084.359 s	0.528 s	
HF_04_0_vorkommende_Zeilen	749599	4544.158 s	2457.162 s	
ismember	749891	2087.012 s	866.169 s	
ismember>ismemberR2012a	749891	1220.843 s	1107.383 s	
HF_13_Solids_bestimmen	21	527.332 s	104.200 s	
F03_3_0_InOutlets	1	484.018 s	328.180 s	
F04_0_Ausgabe	1	465.575 s	0.053 s	
sub2ind	54817710	424.074 s	424.074 s	
HF_12_Randvoxel_bestimmen	3	364.276 s	0.465 s	
F02_0_Rovinge	1	324.375 s	0.016 s	
F02_3_0_Umformen	1	297.240 s	0.002 s	
F02_3_3_0_Umformen2	1	181.904 s	0.140 s	
F02_3_3_1_0_Verschiebungsfunktion	156	181.754 s	1.811 s	
...3_1_2_0_Verschiebungsfunktion_Scheibe	3507	179.818 s	8.968 s	
F04_2_0_Shells_generieren	1	165.499 s	0.304 s	
F02_3_3_1_2_1_freie_Voxel_funktion	16666	160.552 s	44.955 s	
F04_1_0_Solids_generieren	1	157.643 s	0.160 s	
F04_5_Ausgabe_Datei	1	141.304 s	141.304 s	
ismember>ismemberBuiltInTypes	249547	113.460 s	113.460 s	
F02_3_1_0_Kollisionskoerper	1	62.192 s	0.057 s	
F02_3_1_1_Koerper_ermitteln	17	50.434 s	22.915 s	
HF_02_0_Extrudieren	2	49.730 s	0.489 s	

1.1 NUTZEN DES GEORDNETEN GITTERS

Profile Summary

Generated 25-Sep-2016 18:01:22 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Geordnetes_Gitter	1	40.455 s	30.007 s	
ismember	76970	10.399 s	0.897 s	
ismember>ismemberR2012a	76970	9.502 s	2.023 s	
ismember>ismemberBuiltInTypes	76970	7.479 s	7.479 s	
unique	1	0.049 s	0.007 s	
unique>uniqueR2012a	1	0.042 s	0.014 s	
sortrows	1	0.028 s	0.005 s	
sortrows>sort_back_to_front	1	0.023 s	0.023 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

Profile Summary

Generated 25-Sep-2016 18:59:58 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Geordnetes_Gitter2	1	1.399 s	0.343 s	
sub2ind	80000	1.035 s	1.035 s	
unique	1	0.012 s	0.002 s	
unique>uniqueR2012a	1	0.010 s	0.010 s	
ind2sub	1	0.009 s	0.009 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

1.2 INDEXIERUNG

Profile Summary

Generated 25-Sep-2016 19:49:55 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Indexierung1	1	688.500 s	10.543 s	
ismember	10000	677.957 s	3.058 s	
ismember>ismemberR2012a	10000	674.899 s	49.571 s	
sortrows	30000	550.474 s	43.404 s	
sortrows>sort_back_to_front	30000	507.070 s	507.070 s	
unique	20000	365.020 s	2.765 s	
unique>uniqueR2012a	20000	362.255 s	70.945 s	
ismember>ismemberBuiltInTypes	10000	1.143 s	1.143 s	

Self time is the time spent in a function excluding the time spent in its child functions.

Self time also includes overhead resulting from the process of profiling.

Profile Summary

Generated 25-Sep-2016 19:55:27 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Indexierung2	1	3.454 s	3.449 s	
sub2ind	1	0.005 s	0.005 s	

Self time is the time spent in a function excluding the time spent in its child functions.

Self time also includes overhead resulting from the process of profiling.

1.3 VERNÜPFUNG DER VOXEL-ID MIT DER ZEILE

Profile Summary

Generated 25-Sep-2016 20:11:39 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
ListeZeile1	1	3.400 s	3.360 s	
sortrows	1	0.040 s	0.040 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

Profile Summary

Generated 25-Sep-2016 20:13:43 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
ListeZeile2	1	0.043 s	0.006 s	
sortrows	1	0.037 s	0.037 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

1.4 PARALLELISIERUNG

Profile Summary

Generated 25-Sep-2016 20:15:26 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Parallelisierung1	1	0.263 s	0.263 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

Profile Summary

Generated 25-Sep-2016 20:17:33 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Parallelisierung2	1	0.003 s	0.003 s	

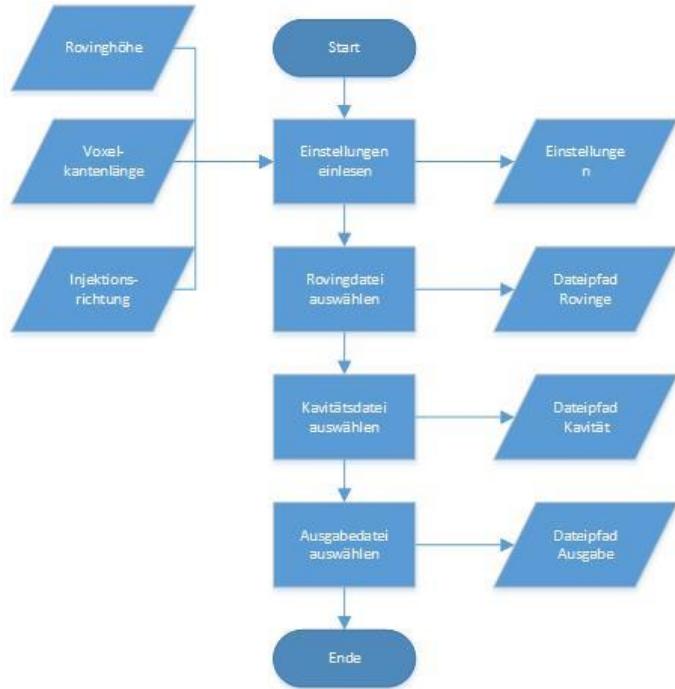
Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

2 Funktionsdiagramme

2.1 F00_MAIN



2.2 F01_EINSTELLUNGEN



2.3 F02_ROVINGE



2.4 F02_1_ROVINGE_TRENNEN



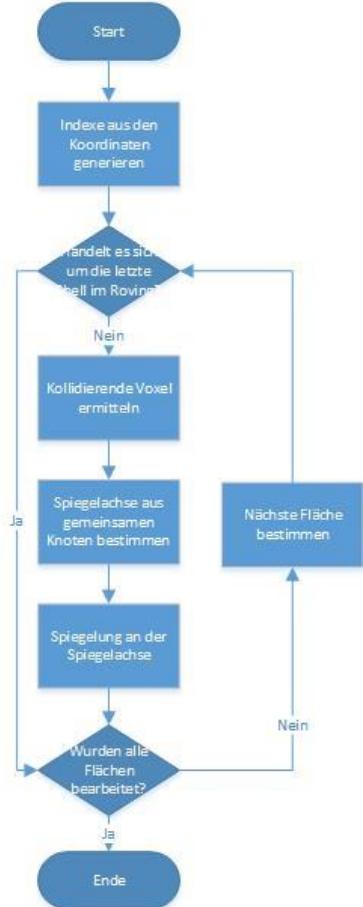
2.5 F02_1_2_0_ZUORDNUNG



2.6 F02_1_3 LETZTE_FLAECHE



2.7 F02_2_0_UEBERGAENGE



2.8 F02_2_2_SPIEGELUNG



2.9 F02_3_0_UMFORMEN



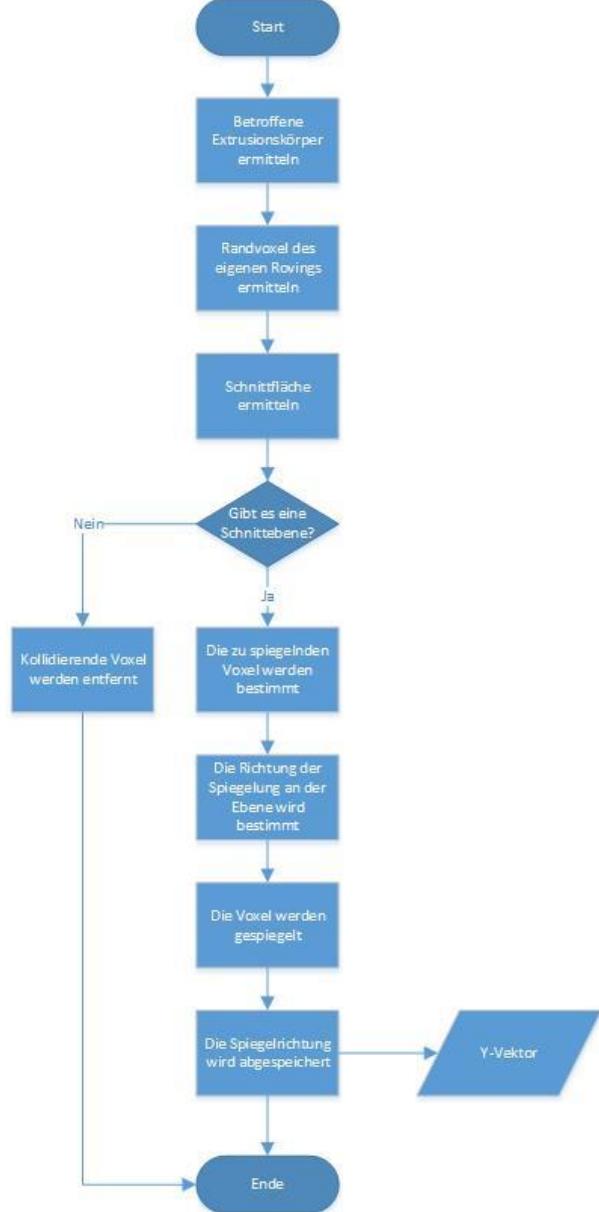
2.10 F02_3_1_0_KOLLISIONSKOERPER



2.11 F02_3_2_0_UMFORMEN_SENKRECHT



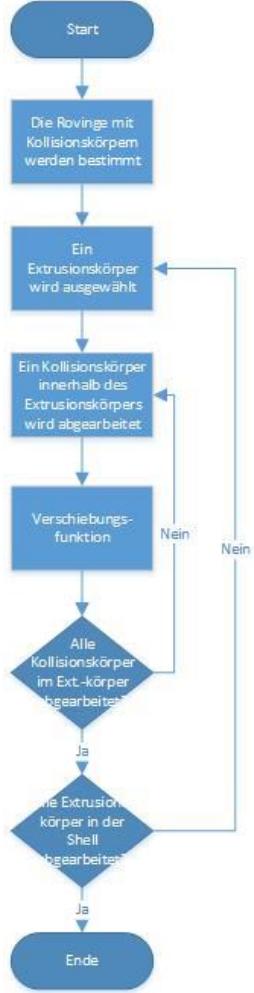
2.12 F02_3_2_1_0_SPIEGELFUNKTION



2.13 F02_3_2_1_3_0_SCHNITTFLAECHE_ERMITTeln



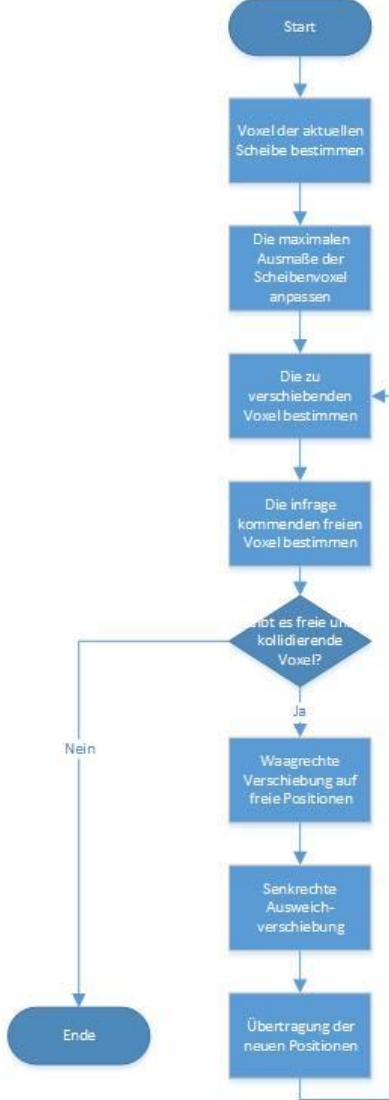
2.14 F02_3_3_0_UMFORMEN_WAAGRECHT



2.15 F02_3_3_1_0_VERSCHIEBUNGSFUNKTION



2.16 F02_3_3_1_2_0_VERSCHIEBUNGSFUNK_SCHEIBE



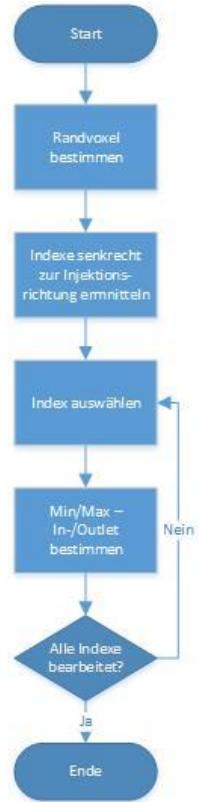
2.17 F03_0_KAVITAET



2.18 F03_2_0_KAVITAET_FUELLEN



2.19 F03_3_0_INOUTLETS



2.20 F04_0_AUSGABE



2.21 F04_1_0_SOLIDS_GENERIEREN



2.22 F04_2_0_SHELLS_GENERIEREN



2.23 F04_2_2_1_SHELLS_FILTERN



2.24 F04_3_KNOTEN_GENERIEREN



2.25 F04_5_AUSGABE_DATEI



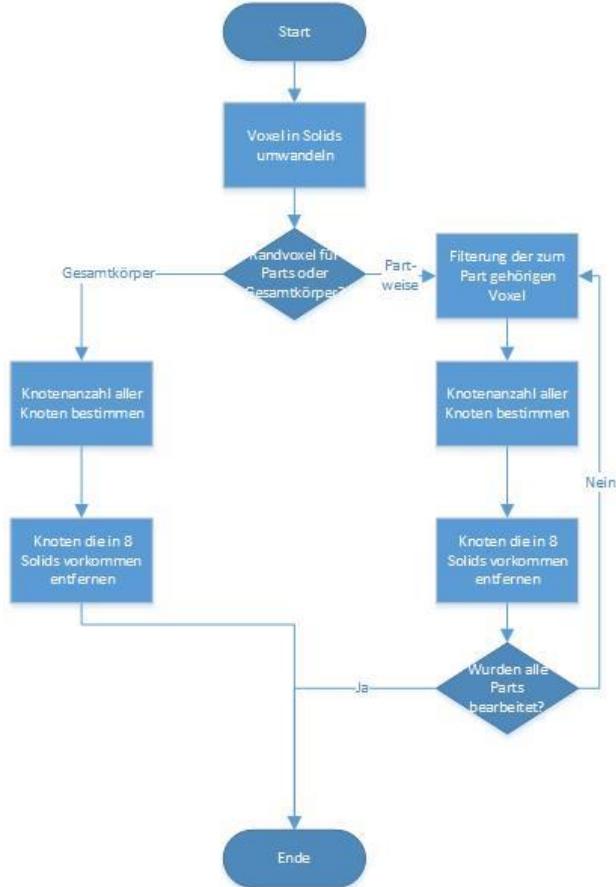
2.26 HF_02_0_EXTRUDIEREN



2.27 HF_06_0_AUFFUELLEN



2.28 HF_12_RANDVOXEL_BESTIMMEN



EIDESSTATTLICHE ERKLÄRUNG

Erklärung

Hiermit versichere ich,

1. dass ich die Arbeit bzw. bei einer Gruppenarbeit den entsprechend gekennzeichneten Anteil der Arbeit selbständig verfasst habe,
2. dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe,
3. dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist,
4. dass die Arbeit weder vollständig noch in Teilen ohne Zustimmung der Prüferin bzw. des Prüfers bereits veröffentlicht worden ist und
5. dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

Statement of Originality

This thesis has been performed independently with support by my supervisors. It contains no material that has been accepted for the award of a degree in this or any other university. To the best of the candidate's knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text. The electronic version is identical to the printed versions.

Datum

Unterschrift

LITERATURVERZEICHNIS

- [1] BMW AG: *BMW 7er Limousine: Fahrdynamik & Effizienz*. URL <http://www.bmw.de/de/neufahrzeuge/7er/limousine/2015/fahrdynamik-effizienz.html#carbon> – Überprüfungsdatum 2016-11-27
- [2] MATZNER, Angelika: *EuroCarBody Award 2015 für die Karosserie mit Carbon Core der neuen BMW 7er Reihe*. 12. November 2015
- [3] EBEL, Bernhard (Hrsg.); HOFER, Markus B. (Hrsg.): *Automotive Management : Strategie und Marketing in der Automobilwirtschaft*. 2., überarb. und aktualisierte Aufl. Berlin : Springer Gabler, 2014
- [4] SPICHER, Ulrich: Kraftstoffverbrauch. In: VAN BASSHUYSEN, Richard (Hrsg.): *Ottomotor mit Direkteinspritzung*. Wiesbaden : Springer Fachmedien Wiesbaden, 2013, S. 189–204
- [5] TECKLENBURG, Gerhard: *Karosseriebau Haltung Hamburg*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014
- [6] HUBER, Ulrich ; STEFFENS, Markus: *CFK — Chancen und Herausforderungen für den Leichtbau*. In: *Lightweight Design* 6 (2013), Nr. 4, S. 16–19
- [7] KLEIN, Bernd: *Leichtbau-Konstruktion : Berechnungsgrundlagen und Gestaltung ; mit Tabellen sowie umfangreichen Übungsaufgaben zu allen Kapiteln des Lehrbuchs*. 10., überarb. und erw. Aufl. Wiesbaden : Springer Vieweg, 2013 (Springer Lehrbuch)
- [8] HENNINGSEN, Michael: *Matrixsysteme für den Leichtbau der Zukunft*. In: *Lightweight Design* 5 (2012), Nr. 4, S. 32–37
- [9] CHERIF, Chokri (Hrsg.): *Textile Werkstoffe für den Leichtbau : Techniken - Verfahren - Materialien - Eigenschaften*. Berlin : Springer, 2011
- [10] ZHU, Hongyan ; WU, Baochang ; LI, Dihong ; ZHANG, Dongxing ; CHEN, Yuyong: *Influence of Voids on the Tensile Performance of Carbon/epoxy Fabric Laminates*. In: *Journal of Materials Science & Technology* 27 (2011), Nr. 1, S. 69–73
- [11] RODRIGUEZ, E. ; GIACOMELLI, F. ; VAZQUEZ, A.: *Permeability-Porosity Relationship in RTM for Different Fiberglass and Natural Reinforcements*. In: *Journal of Composite Materials* 38 (2004), Nr. 3, S. 259–268
- [12] MIDENDORF, Peter ; MICHAELIS, Daniel ; BÖHLER, P. ; DITTMANN, J. ; HEIECK, F.; ARENA2036 – DigitPro: Development of a virtual process chain. In: BARGENDE, Michael; REUSS, Hans-Christian; WIEDEMANN, Jochen (Hrsg.): *16. Internationales Stuttgarter Symposium*. Wiesbaden : Springer Fachmedien Wiesbaden, 2016 (Proceedings), S. 505–516
- [13] LOMOV, S. ; IVANOV, D. ; VERPOEST, I. ; ZAKO, M. ; KURASHIKI, T. ; NAKAI, H. ; HIROSAWA, S.: *Meso-FE modelling of textile composites : Road map, dataflow and algorithms*. In: *Composites Science and Technology* 67 (2007), Nr. 9, S. 1870–1891
- [14] J.SIRTAUTAS: *Braiding simulation and mesomechanical modeling of braided composites*. Stuttgart, University of Stuttgart, Institute of Aircraft Design. Master Thesis. 2009
- [15] THE 8TH INTERNATIONAL CONFERENCE ON FLOW PROCESSES IN COMPOSITE MATERIALS (Hrsg.): *WISETEX-BASED MODELS OF PERMEABILITY OF TEXTILES*, 2006
- [16] VEIT, Dieter (Hrsg.): *Simulation in textile technology : Theory and applications*. Oxford : Woodhead Pub, 2012 (Woodhead Publishing series in textiles no. 136)
- [17] LOMOV, S. V. ; T.PEETERS: *INTEGRATED TEXTILE PREPROCESSOR WISETEX : Computational models, methods and algorithms*. Leuven, KU Leuven. VERSION 2.3. 2002
- [18] LOMOV, S. V. ; A.N.MOGILNY ; NEVSKAYA MANUFACTURE: *Mathematical modelling of porosity of plane and 3D woven structures*. In: *Transactions on Engineering Sciences* 1998 (1998), Nr. 21

- [19] LOMOV, S. V.: *MODELLING THE GEOMETRY OF TEXTILE COMPOSITE REINFORCMENTS*: WISETEX. Leuven, KU Leuven, Department MTM. URL <HTTP://WWW.MTM.KULEUVEN.BE/ONDERZOEK/COMPOSITES/COMPOSIETEN>
- [20] LIN, H. ; ZENG, X. ; SHERBURN, M. ; LONG, A. C. ; CLIFFORD, M. J.: *Automated geometric modelling of textile structures*. In: *Textile Research Journal* 82 (2012), Nr. 16, S. 1689–1702
- [21] SHERBURN, Martin: *Geometric and Mechanical Modelling of Textiles*. Nottingham, University of Nottingham. PhD Thesis. Juli 2007. URL <http://eprints.nottingham.ac.uk/10303/1/thesis-final.pdf>
- [22] 1ST WORLD CONFERENCE ON 3D FABRICS (Hrsg.): *MODELLING 3D FABRICS AND 3D-REINFORCED COMPOSITES: CHALLENGES AND SOLUTIONS*, 2008
- [23] M. FOUINNETEAU: *Damage and failure modelling of carbon and glass 2D braided composites*. CRANFIELD, CRANFIELD UNIVERSITY. PhD Thesis. April 2006
- [24] LOMOV, S. V. ; VERPOEST, I. ; CICHOSZ, J. ; HAHN, C. ; IVANOV, D. S. ; VERLEYE, B.: *Meso-level textile composites simulations : Open data exchange and scripting*. In: *Journal of Composite Materials* 48 (2014), Nr. 5, S. 621–637
- [25] S.D. GREEN ; M.Y. MATVEEV ; A.C. LONG ; D. IVANOV ; S.R. HALLETT: *Mechanical modelling of 3D woven composites considering realistic unit cell geometry*
- [26] LOMOV, S. V. ; HUYSMANS, G. ; LUO, Y. ; PARNAS, R. S. ; PRODROMOU, A. ; VERPOEST, I. ; PHELAN, F. R.: *Textile composites : Modelling strategies*. In: *Composites Part A: Applied Science and Manufacturing* 32 (2001), Nr. 10, S. 1379–1394
- [27] VERLEYE, B.: *COMPUTATION OF THE PERMEABILITY OF MULTI-SCALE POROUS MEDIA WITH APPLICATION TO TECHNICAL TEXTILES*. Leuven, KU Leuven. Phd Thesis. März 2008
- [28] NGUYEN, Q. T. ; VIDAL-SALLÉ, E. ; BOISSE, P. ; PARK, C. H. ; SAOUAB, A. ; BRÉARD, J. ; HIVET, G.: *Mesoscopic scale analyses of textile composite reinforcement compaction*. In: *Composites Part B: Engineering* 44 (2013), Nr. 1, S. 231–241
- [29] BADEL, P. ; VIDAL-SALLÉ, E. ; MAIRE, E. ; BOISSE, P.: *Simulation and tomography analysis of textile composite reinforcement deformation at the mesoscopic scale*. In: *Composites Science and Technology* 68 (2008), Nr. 12, S. 2433–2440
- [30] CAWKELL, A. E.: *Encyclopaedic dictionary of information technology and systems*. London : Bowker Saur, 1993
- [31] CHIDAMBER, S. R. ; KEMERER, C. F.: *A metrics suite for object oriented design*. In: *IEEE Transactions on Software Engineering* 20 (1994), Nr. 6, S. 476–493
- [32] V.MENON ; A. E. TREFETHEN: MultiMATLAB Integrating MATLAB with High Performance Parallel Computing : Supercomputing. In: *ACM/IEEE 1997 Conference*, S. 30
- [33] POTLURI, P. ; PARLAK, I. ; RAMGULAM, R. ; SAGAR, T. V.: *Analysis of tow deformations in textile preforms subjected to forming forces*. In: *Composites Science and Technology* 66 (2006), Nr. 2, S. 297–305
- [34] CHEN, Baoxing ; CHOU, Tsu-Wei: *Compaction of woven-fabric preforms in liquid composite molding processes : Single-layer deformation*. In: *Composites Science and Technology* 59 (1999), Nr. 10, S. 1519–1526