




Lehrstuhl für Verfahrenstechnik des industriellen Umweltschutzes

Masterarbeit



Strukturbildung und
Durchströmungssimulationen virtueller
Grünmix-Packungen

Andrea Höftberger, BSc

März 2019

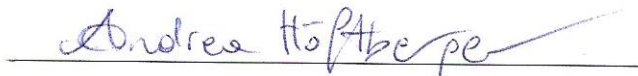
EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 15.03.2019



Unterschrift Verfasser/in
Andrea Höftberger
Matrikelnummer: 01235441

Vorgelegt von:

Andrea Höftberger

m01235441

Betreuer/Gutachter:

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Christian Weiß

DANKSAGUNG

Besonderem Dank gilt meinem Betreuer, Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Christian Weiß. Du hast mir die Möglichkeit gegeben, eine mir neue Materie zu erforschen und dabei viel Geduld bewiesen. Danke, dass ich mich während der turbulenten Entstehungsphase der Arbeit jederzeit bei Fragen oder Problemen an dich wenden konnte. Die viele Zeit, die du mit in dieses Projekt gesteckt hast, hat sich definitiv ausgezahlt. Ich bin stolz auf das Ergebnis und hätte es ohne dich wahrscheinlich nicht in dieser Zeitspanne und Qualität geschafft.

Mein Dank gilt auch meinen Eltern, die mich nicht nur finanziell unterstützt, sondern auch in allen anderen Lebensphasen den Rücken gestärkt haben. Ohne ihre „Essenspakete“ hätte meine Nahrungsaufnahme wahrscheinlich sehr einseitig aus Nudeln und Pesto bestanden. Auch meine Freunde und Kollegen will ich erwähnen, die mir meine Freizeit verschönert oder mich mit ihrer fachlichen Kompetenz unterstützt haben. Abschließend sei mein Partner genannt, der mich während der gesamten Studienzeit begleitet hat. Obwohl er meine Prüfungssorgen schon in und auswendig kannte, hat er mir jedes Mal wieder sein Ohr geborgt und ist mir Rat und Tat zur Seite gestanden.

Kurzfassung

Strukturbildung und Durchströmungssimulationen virtueller Grünmix-Packungen

Im Sinterprozess ist neben der Qualität des fertigen Sinters auch die Sinterleistung wesentlich, für welche die Durchgasbarkeit als einer der Schlüsselfaktoren angesehen wird. Zum spezifischen Druckverlust bzw. der Permeabilität beim Durchströmen liegen bereits zahlreiche experimentelle Untersuchungsergebnisse vor. In dieser Arbeit wird versucht, einen neuen, numerischen Zugang zu dieser Thematik zu finden, indem virtuelle Modellpackungen mittels eines Materialsimulators GeoDict erstellt werden. Im Mittelpunkt stehen dabei unterschiedliche Methoden zur Generierung einer möglichst dichten Kornpackung, welche einen kleinen, repräsentativen Ausschnitt einer Schüttung aus dem Sinterrohmaterialien darstellen sollen. Als weitere Möglichkeit, um an eine virtuelle, dichte Packung zu gelangen, wird von einem CT-Scan eines fertigen Sinterstücks ausgegangen. Die geometrischen Strukturparameter der Packungen werden analytisch und numerisch ermittelt. Im Folgeschritt wird die Porenströmung in den Packungsmodellen numerisch berechnet, um wichtige Kennparameter, wie den spezifischen Druckverlust und die Permeabilität zu quantifizieren. Die Ergebnisse sind schließlich Grundlage für vergleichende Gegenüberstellungen mit experimentellen Daten und empirischen Druckverlustgleichungen.

Abstract

Generation and porous flow simulation of virtual granular packings

For the sinter process the productivity is as important as the quality of the sinter and thereby bed or packing permeability is seen as one of the key factors. Numerous prior studies experimentally investigated permeability and pressure drop of sinter raw-mix packings. This thesis offers a new approach to the topic based on numerical methodology by application of the digital material laboratory GeoDict for generation of virtual packings of free formed particles. Therefore, different kinds of methods are explored to generate small representative volumes with realistic granular packing structure. Alternatively in a top-down approach, a CT-scan of product sinter was used as a starting point for the generation of ideal dense packings. Structural parameters of the virtual packings were characterized by a combined set of analytical and numerical methods. In the follow-up step the porous gas flow in the virtual model packings is investigated by numerical flow simulations. Permeability and specific pressure drop are evaluated as characteristic parameters. The numerical results are then compared to experimental data and empirical model equations.

Inhaltsverzeichnis

	Seite
1 EINLEITUNG	3
1.1 Begriffsdefinition	3
1.2 Problemstellung und Zielsetzung	5
2 PACKUNGSGENERIERUNGEN	6
2.1 Sphärische Modellpackung mit Korngrößenverteilung gemäß RRSB	8
2.1.1 Herkunft der verwendeten Korngrößenanalyse	8
2.1.2 Korngrößenverteilung und Kornklassen aus der Analyse	9
2.1.3 Wahl eines geeigneten analytischen Verteilungsgesetzes	10
2.1.4 Spezifische Oberfläche und Sauterdurchmesser	12
2.1.5 Sphärizität und Formfaktor	13
2.1.6 Porosität	15
2.1.7 Approximierte Verteilung basierend auf den RRSB-Parametern	16
2.1.8 Präsentation der erstellten sphärischen Packung	18
2.2 Mehreckige Modellpackung mit Korngrößenverteilung gemäß RRSB	19
2.2.1 Generierung der Kornformgruppen	19
2.2.2 Ansätze zur Erstellung einer dichten Packung	23
2.2.3 Präsentation der erstellten polyedrischen Packung	29
2.3 Freie Positionierung von skalierten Polyedern	30
2.3.1 Neue Korngrößenverteilung	30
2.3.2 Erstellung der Modellkörner	32
2.3.3 Prinzip bei der Packungserstellung	33
2.3.4 Präsentation der fertigen Packung mit Kennparametern	42
2.4 CT-basierte Sinterstruktur	43
2.4.1 Modifikation des Sinters	44
2.5 Übersicht der Modelle	45
3 NUMERISCHE ERMITTLUNG VON PACKUNGSEIGENSCHAFTEN	47
3.1 Geometrische Strukturparameter	47
3.1.1 Numerische Ermittlungsmöglichkeiten	47
3.2 Berechnungsmodelle für die Strömungssimulation	50
3.2.1 Randbedingungen und Eigenschaften des Lösungsalgorithmus	52
3.3 Empirische Druckverlustgleichungen und Permeabilität	55
4 DISKUSSION DER ERZEUGTEN PACKUNGSSTRUKTUREN	59

4.1	Vergleich der Korngrößenverteilungen und Strukturparameter.....	59
4.2	Vergleich der Modellpackungen.....	62
5	VERGLEICHENDE DISKUSSION DER DURCHSTRÖMUNGS- CHARAKTERISTIK	63
5.1	Ergebnisse aus der Strömungssimulation für Packung „Kugel“	63
5.2	Ergebnisse aus der Strömungssimulation der Packung „Polyeder-46“	68
5.2.1	Variation der Packung „Polyeder-46“ durch Entfernen der kleinsten Kornformgruppe.....	71
5.3	Ergebnisse der Strömungssimulation der Packung „Polyeder-36“	74
5.4	Druckverlustcharakteristik der CT-basierten Sinterstruktur	77
6	ZUSAMMENFASSUNG	80
7	VERZEICHNISSE	81
7.1	Literatur.....	81
7.2	Abkürzungsverzeichnis	82
ANHANG	I

1 Einleitung

1.1 Begriffsdefinition

Im folgenden Kapitel werden die Definition und Grundbegriffe, welche mit dem Thema der Arbeit im Zusammenhang stehen, erklärt. In der Roheisenherstellung zählt der Hochofen zu den bedeutendsten Verfahren. Während des Prozesses wird in das Aggregat von unten heiße Luft eingeblasen, während er von oben abwechselnd mit Möller und Koks beschickt wird. Beim Möller handelt es sich um ein Mischmaterial aus eisenerzhaltigen Stoffen. Um eine gute Durchströmbarkeit zu gewährleisten, muss das eingesetzte Material stückig bzw. porös sein, ansonsten könnte die hohe Temperatur im Inneren nicht gehalten werden. Diese Anforderung, sowie eine gute Reduzierbarkeit und ein hoher Eisengehalt erfordern laut [2] eine Aufbereitung der Erze. Neben der Zerkleinerung und dem Sieben beinhaltet dieses unter anderem das Sintern feiner Eisenerze, die beim Brechen des Materials entstehen und bei direktem Einsatz den Hochofen verstopfen würden.

Unter dem Sintern wird ein Aufbereitungsprozess verstanden, bei dem eisenhaltige Stoffe, Zuschlagsstoffe und Koks durch thermische Behandlung agglomerieren. Während des Prozesses werden unerwünschte Begleitstoffe, wie CO_2 oder Kristallwasser, entfernt und das Erz somit angereichert. Außerdem führt nach [21] das Sintern zu einer Oxidation des Eisens zu seiner höchsten Oxidationsstufe Fe_2O_3 . Das eingesetzte Feinerz muss einigen Anforderungen entsprechen, so ist neben seiner chemischen Zusammensetzung die Konstanz seiner mineralogischen Zusammensetzung wichtig. In seiner Korngrößenverteilung darf es einen Wertebereich zwischen 0,1 und 10 mm besitzen, da zu feine Mischungen die Durchgasung behindern und zu grobe Stücke eine schlechte Wärmeübertragung aufweisen. Nebenstoffe, wie zum Beispiel Gichtstaub und Walzzunder, welche im Laufe des Hüttenkreislaufes entstehen, können zusätzlich mitverarbeitet werden; [2]. Diese landen somit nicht auf der Deponie und der Kohlenstoffgehalt im Gichtstaub hilft beim Einsparen von Brennstoff. Die Zuschlagsstoffe im Sinter sind meist basisch (z.B. CaCO_3 , CaO), um eine bestimmte Basizität später in der im Hochofen anfallenden Schlacke zu erreichen. Der Koks dient als Brennstoff, dessen Hitze die Oberfläche des Gemisches aufschmilzt und zum Kleben der Partikel führt. Als Grünmix wird hierbei das fertige Gemisch aus den bereits genannten Stoffen genannt, bevor dieses der thermischen Behandlung unterzogen werden. Abbildung 1 stellt das komplette Fließschema der Sinteranlage dar inklusive des Herstellungsweges des Grünmixes, welcher in [2] beschrieben ist. Die Einsatzmaterialien Kalk, Erz und Brennstoff (1) werden zuerst über Dosierbunker (3) auf eine Mischbettenanlage (5) gebracht, von wo diese zur Misch- und Rolliertrommel (7) gelangen. Zur Nachdosierung von Zuschlägen, Brennstoff und zur Zugabe des Rückgutes und anderen Nebenstoffen sind nochmals Dosierbunker (6) vorgeschaltet. In der Mischtrommel wird Wasser zugegeben, welches dabei hilft, in der Rolliertrommel die feinen Partikel an die größeren zu agglomerieren. Diese Vorbereitung ist nötig, bevor der Grünmix gesintert werden kann.

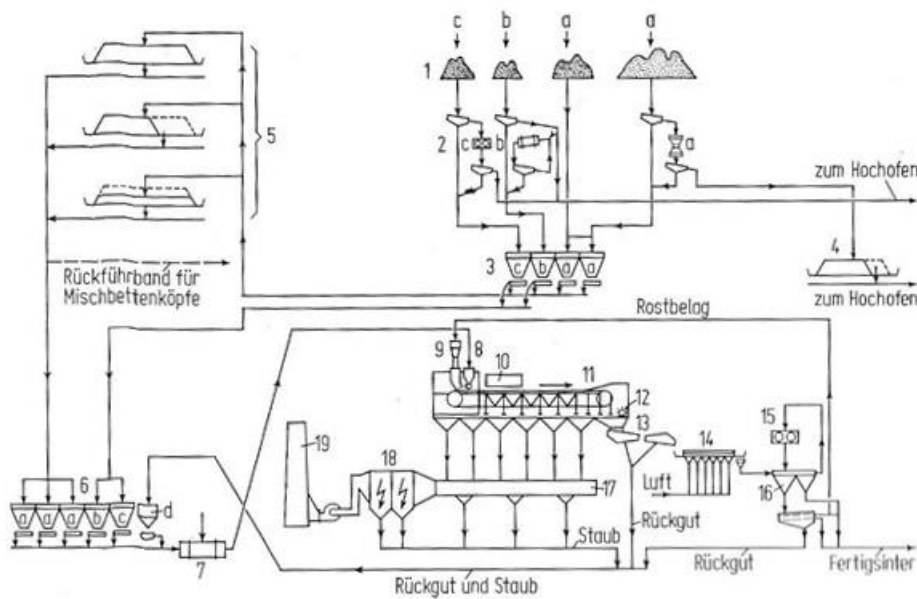


Abbildung 1: Fließschema einer Sinteranlage [2]

Der Sinterprozess selbst wird üblicherweise auf einer kontinuierlichen Bandsintermaschine durchgeführt, wobei eine detaillierte Darstellung in Abbildung 2 illustriert ist. Laut [21] wird zu Beginn auf den Wanderrost eine Schutzschicht mit 3 bis 5 cm aus bereits gesintertem Material aufgetragen. Danach folgt die eigentliche Aufgabe des Grünmixes mit einer Schichthöhe von bis zu 700 mm. Unter der Zündhaube erfolgt die Zündung des Brennstoffes und durch das Ansaugen von Luft über Saugzugkästen unter dem Rost wandert die Brennschicht langsam nach unten. Am Ende des Bandes fällt der fertige Sinter in einen Brecher mit anschließender Siebung. Das kalte Rückgut mit einer Korngröße < 5 mm wird dem Grünmix zugeführt, während jenes Material mit einer Korngröße zwischen 10 und 25 mm als Rostbelag verwendet wird. Der Fertigsinter besteht typischer Weise aus Stücke mit 5 bis 50 mm.

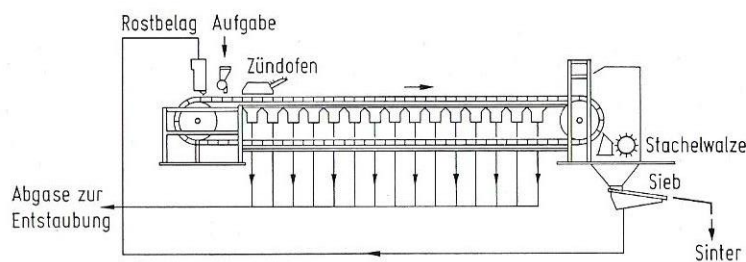


Abbildung 2: Schematische Darstellung einer Sintermaschine [2]

Derzeit laufende Studien zur Optimierung des Sinterprozesses befassen sich unter anderem mit Möglichkeiten zur Verringerung des Koksbedarfes und deren Auswirkungen auf die Abgaszusammensetzung. Die gegenständliche Arbeit ist eingebunden in das Sinteroptimierungsprojekt und fokussiert sich auf die numerische Untersuchung der Permeabilität und damit zusammenhängender Parameter. Der Grünmix wird als binäre,

virtuelle Struktur dargestellt und somit für computergestützte Folgeuntersuchungen zugänglich gemacht, wobei die zu untersuchende Packung geschüttet vorliegt.

1.2 Problemstellung und Zielsetzung

Die Permeabilität wird als Schlüsselfaktor in der Praxis zur optimalen Sinterherstellung gesehen. Messschwierigkeiten vor Ort am Band, um ihre Einflussfaktoren systematisch zu bestimmen, stellten große Herausforderungen dar. Fehlende theoretische Kenntnisse konnten bisher durch lange Erfahrung ausgeglichen werden. Entwicklungen in der Simulation ermöglichen jetzt eine numerische Herangehensweise, mit der ein besseres Verständnis für die Bedeutung der Grünmix-Permeabilität beim Sinterprozess gewonnen werden kann. Virtuelle Modelle der Packungsstruktur, auf denen die numerischen Simulationen basieren, haben den Vorteil, dass sie relativ einfach in ihrer Struktur und anderen Parametern modifiziert werden können.

In dieser Arbeit werden virtuelle Modelle einer Grünmix-Schüttung mithilfe der Simulationssoftware GeoDict erzeugt und in weiterer Folge durchströmt. Dabei werden physikalische Eigenschaften, wie Dichte, Härte, etc. vernachlässigt und nur die Partikelform und -verteilung berücksichtigt.¹⁾ Insgesamt werden drei verschiedene Modelle generiert, welche sich in der Kornform, Porosität und Korngrößenverteilung unterscheiden. Des Weiteren variiert die Herangehensweise für die Generierung der unterschiedlichen Strukturmodelle. Neben „bottom-up“ Strategien der Packungsgenerierung durch räumliche Anordnung vorgefertigter Einzelpartikel wird alternativ auch eine „top-down“ Methode eingesetzt, bei der von einem CT-Scan eines Fertigsinterstücks als Startpunkt für die Strukturzeugung ausgegangen wird. Dies stellt in Kombination mit Strukturwandlungsmethoden (ab- bzw. aufbauend) eine andere Möglichkeit dar, an eine dichte, virtuelle Packung zu gelangen. Basierend auf diesen Modellen werden numerische Strömungssimulationen durchgeführt und Druckverlustkurven berechnet. Diese werden wiederum mit experimentellen Daten und mathematischen Druckverlust-Modellen verglichen. In weiterer Folge werden die unterschiedlichen Berechnungsmethoden für die spezifische Oberfläche diskutiert, weil diese einen entscheidenden Faktor in den Druckverlust-Modellen einnimmt.

1) Die Software erlaubt aber, den Packungselementen im Nachhinein ein Material zuzuordnen.

2 Packungsgenerierungen

In diesem Kapitel werden die unterschiedlichen Methoden zur Generierung von Grünmix-Packungen vorgestellt. Insgesamt werden vier unterschiedliche Varianten entwickelt, welche ein vereinfachtes Abbild der realen Schüttung darstellen sollen. Simulationen basierend auf diesen sollen in späterer Folge mit experimentellen Daten und mathematischen Gleichungen verglichen werden.

Der reale Grünmix besitzt eine sehr komplexe Morphologie, das heißt seine Kornform lässt sich nicht durch einfache geometrische Formen, wie zum Beispiel Würfel, Quader oder Kugel, beschreiben. Die Körner können von stabförmigen, über pyramidenförmigen, elliptischen bis hin zu sphärischen Figuren reichen. Des Weiteren ist die Oberfläche der Partikel polyedrisch aufgebaut und verfügt über Anpackungen unterschiedlicher Größe, welche durch die Feuchte im Schüttgut an den Körnern kleben bleiben. Abbildung 3 zeigt anhand einiger exemplarisch ausgewählter Partikel die komplizierte Kornform.



Abbildung 3: Einzelne Partikel aus einer realen Grünmix-Schüttung zur Veranschaulichung der morphologischen Komplexität

Zusätzlich zur Untersuchung einzelner Körner wird eine Grünmix-Probe in ein Becherglas mit einem Fassungsvermögen von 600 mL und einem Durchmesser von 10cm geschüttet. In Abbildung 4 sind zwei unterschiedliche Seiten der entstehenden Schüttung auf gleicher Höhe dargestellt. Es ergeben sich dicht gepackte Bereiche mit einem hohen Feintanteil und sehr poröse Bereiche, verursacht durch die größeren, unregelmäßig geformten Körner. Die schwierig zu beschreibende Kornform, als auch die unregelmäßige Verteilung in der Schüttung bedingen eine starke Vereinfachung bei der Übertragung der realen Morphologie in die virtuelle Welt.

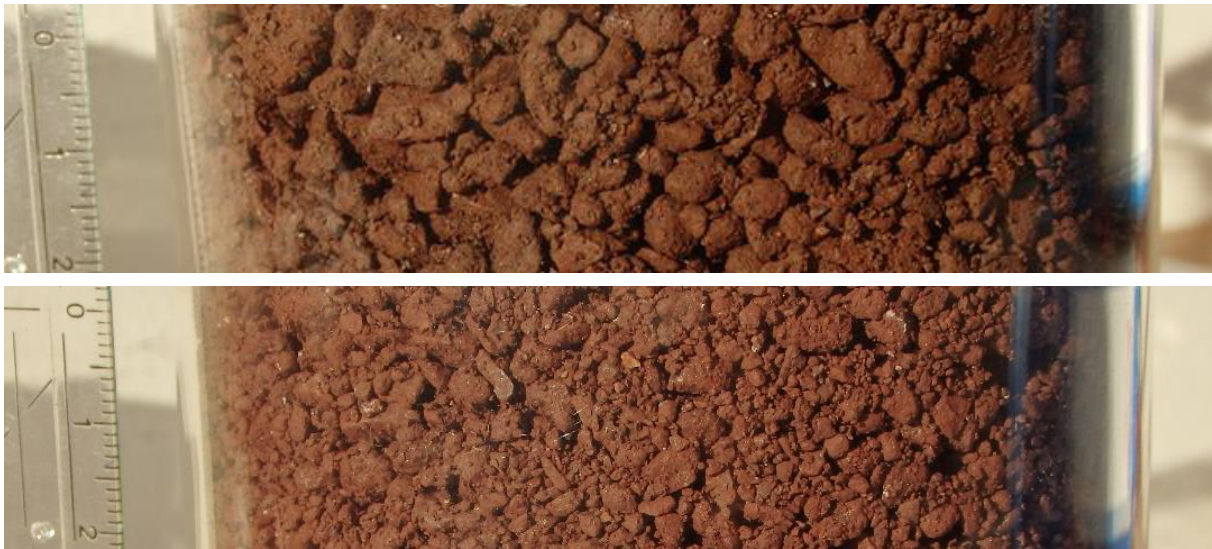


Abbildung 4: Zwei exemplarische Ausschnitte eines geschüttet Grünmixes in einem Becherglas

Die Struktur der virtuellen, dreidimensionalen Grünmix-Packung ist durch würfelförmige Voxel aufgebaut. Diese stellen das kleinste Einheitselement der Packungsstruktur dar. In der nachfolgenden Abbildung 5 ist zum besseren Verständnis eine Strukturoberfläche in Nahaufnahme dargestellt und rechts daneben ein einzelner Voxel. Seine Seitenlänge im Verhältnis zu der Größe der Modellbox bestimmt die Auflösung der darin befindlichen Partikel. Je kleiner die Voxel in Relation zur Kantenlänge der Box sind, desto glatter ist die Partikeloberfläche. Begrenzt wird das Verhältnis durch die Rechenleistung bzw. –zeit. Aus demselben Grund kann die Packung nicht für die gesamte Schüttung modelliert werden, sondern es wird ein repräsentativer Ausschnitt gewählt. Dieser stellt die größtmögliche Box dar, um trotzdem die Korngrößenverteilung, Schüttdichte, Druckänderungen etc. wiederzugeben. Ein Voxel ist somit das Basiselement zur repräsentativen Darstellung der gesamten Schüttung.

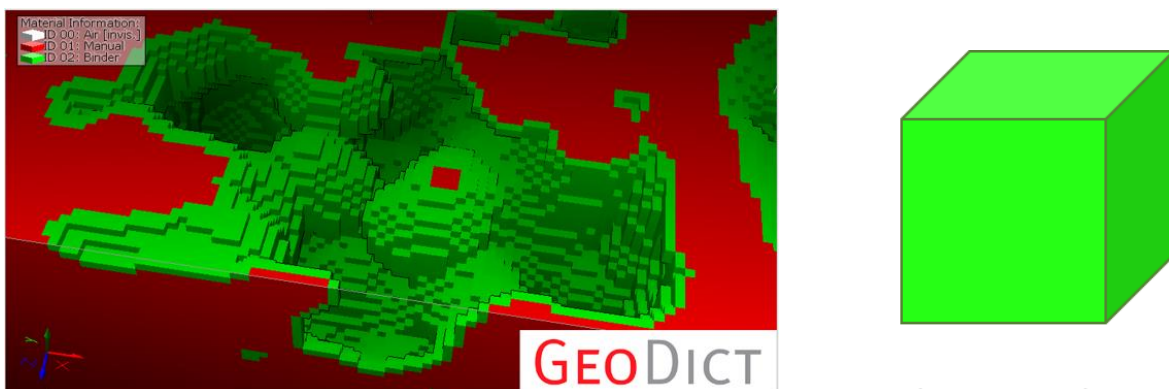


Abbildung 5: Links: Ausschnitt einer Oberflächenstruktur; Rechts: Darstellung eines einzelnen Voxels

2.1 Sphärische Modellpackung mit Korngrößenverteilung gemäß RRSB

Für das erste Modell wird eine einfache Kugelform als wesentliches Strukturmerkmal der Partikel ausgewählt, welche sich gegenseitig in der Schüttung nicht überlappen. Die Korngrößenverteilung basiert dabei auf der Analyse einer realen Grünmix-Schüttung.

2.1.1 Herkunft der verwendeten Korngrößenanalyse

Um eine repräsentative Auswahl zu treffen, werden insgesamt 12 Proben aus zwei verschiedenen Quellen verglichen. Acht Proben stammen aus der Masterarbeit von Anna Griesser ([5]), welche ihre Analysen mit einem HAVER CPA-Messgerät durchführte. Bei der gewählten Methodik handelt es sich um eine dynamische, photooptische Analyse mittels Zeilenkamera. Abbildung 6 illustriert das Funktionsprinzip: In dem Gerät wird die Probe über eine Dosierrinne zwischen einer LED-Lichtquelle und der Kamera transportiert und dort fallen gelassen. Im freien Fall „scant“ die Zeilenkamera die Partikel und erstellt ein zweidimensionales Schattenbild; [6]. Anhand der Daten erstellt die dazugehörige CpaServ Software eine Korngrößenanalyse und berechnet weitere Parameter, wie die spezifische Oberfläche, Sphärizität, etc.

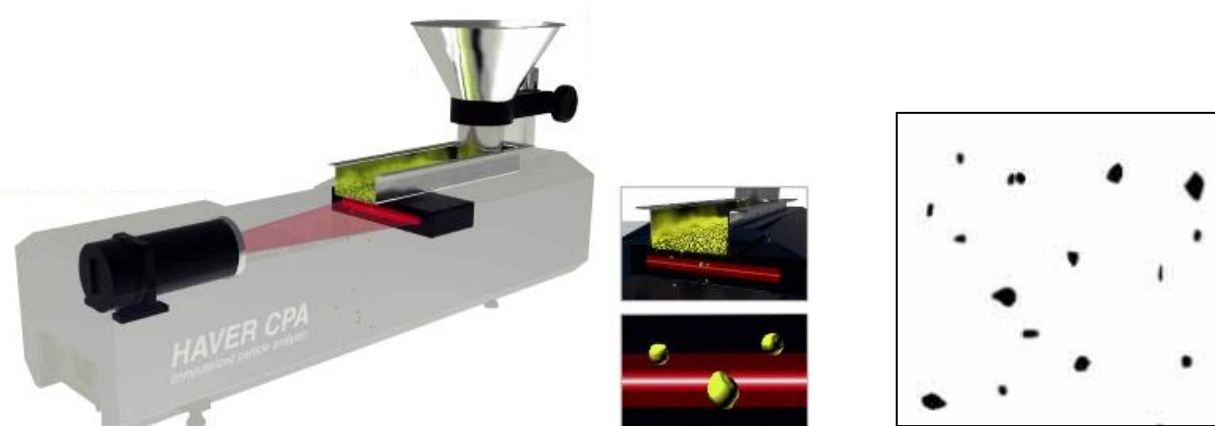


Abbildung 6: Links: Aufbau eines HAVER CPA-Messgerätes; Rechts: Ausschnitt eines bereits gescannten Abschnittes dargestellt in der Softwareoberfläche [5]

Weitere vier Proben werden der Doktorarbeit von Hannes Menapace ([16]) entnommen, welcher seine Analyse mittels Siebmaschine durchführte. Dabei wurden fünf Siebe verwendet, dessen Maschenweiten zwischen 0,5 und 6,3 mm lagen. Im Vergleich zu der gewählten Methode von [5] ergibt sich durch die geringe Siebzahl, sowie der geringen Bandbreite eine verringerte Auflösung der Korngrößeninformationen. Er stellte die Ergebnisse nicht in Tabellenform mit exakten Werten dar, sondern als Summendurchgangskurve bzw. die relative Häufigkeit in einem Balkendiagramm.

Für die Auswahl werden alle Proben miteinander verglichen und nach dem Ausschlussprinzip nacheinander jene Verteilungen mit zu starken Abweichungen ausgeschlossen. Aufgrund der

besseren Konsistenz werden als Basis für die Packungsgenerierung insbesondere die Basisdaten aus [5] herangezogen.

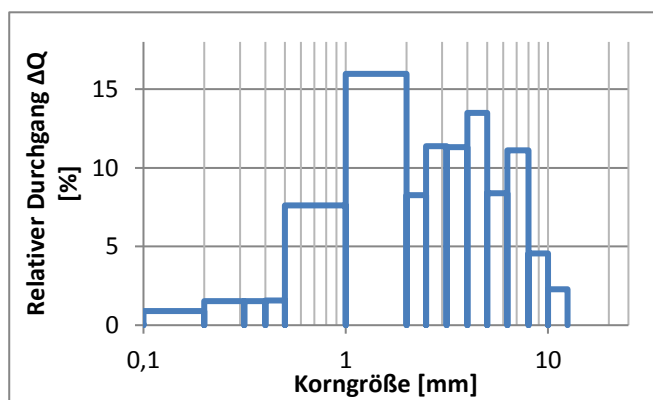
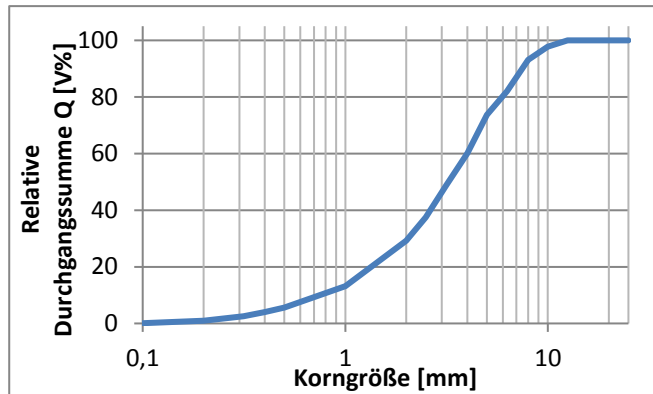
2.1.2 Korngrößenverteilung und Kornklassen aus der Analyse

Als Basis für die Korngrößenverteilung wird schließlich die sechste Probe aus [5] (SV 4-15 Probe 6) gewählt. Wie in Tabelle 1 zu sehen, wird das Analysenergebnis in 14 Kornklassen eingeteilt, welche sich bis zu einer maximalen, oberen Grenze von 12,5 mm erstrecken. Die Breite der Klassen nimmt dabei mit sinkender Korngröße stetig ab. Bei den Ergebnissen handelt es sich um die relative Durchgangssumme Q und dem relativen Durchgang ΔQ , deren Verläufe links neben der Tabelle grafisch aufbereitet sind. Diese Begriffe stammen aus der Korngrößenanalyse mittels Siebung. Als relative Durchgangssumme wird dabei jener Anteil der untersuchten Mengen beschrieben, welcher durch die Siebmaschen eines Netzes mit definierter Maschengröße fällt. Jener Anteil, der auf dem Sieb zurückbleibt wird als Rückstand R bezeichnet. Die Differenz der Durchgangssummen zweier aufeinanderfolgender Kornklassen ergibt den relativen Durchgang ΔQ .

Der relative Durchgang aufsummiert in einem Spektrum von 1 bis 5 mm ergibt einen Wert von 60,14 %. Wie erwartet befinden sich hier also die meisten Partikel. Nach [21] besteht die Körnung des eingesetzten Brennstoffes im Grünmix zum Großteil aus einer Fraktion zwischen 1 und 3 mm und auch die Korngröße der Zuschlagsstoffe befindet sich in einem Bereich <2 mm. Außerdem kommt in [5] jener fertig gesinterte Anteil aus dem Sinterprodukt, welcher die Mindestgröße von 5 mm unterschreitet, zum Rohmix hinzu. Der Feinanteil haftet durch die Wasserzugabe und das anschließende Rollieren an der größeren Partikeln, sodass der freie Feinanteil vergleichsweise gering ausfällt. Auch der Anteil der größeren Stücke (> 6 mm) ist gering, da dieser bereits als Stückerz direkt im Hochofen eingesetzt werden kann und deswegen vorher abgesiebt wird.

Tabelle 1: Links: Korngrößenanalyse aus [5]; Rechts: Logarithmische Darstellung der relativen Durchgangssumme und des relativen Durchgangs der Verteilung

Ergebnis der Korngrößenanalyse		
x [mm]	Q [V%]	ΔQ [V%]
25	100	0
20	100	0
16	100	0
12,5	100	0
10	97,71	2,29
8	93,14	4,57
6,3	82,02	11,12
5	73,64	8,38
4	60,14	13,5
3,15	48,83	11,31
2,5	37,46	11,37
2	29,2	8,26
1	13,23	15,97
0,5	5,62	7,61
0,4	4,04	1,58
0,315	2,51	1,53
0,2	0,99	1,52
0,1	0,09	0,9
<0,1		0,09
Summe		100



2.1.3 Wahl eines geeigneten analytischen Verteilungsgesetzes

Die nachfolgenden Formeln, Definitionen, sowie Beschreibungen in diesem Kapitel 2.1 entstammen dem Buch Partikeltechnologie von Matthias Stieß ([22]). Auf andere Quellen wird direkt bei dem betreffenden Gegenstand verwiesen.

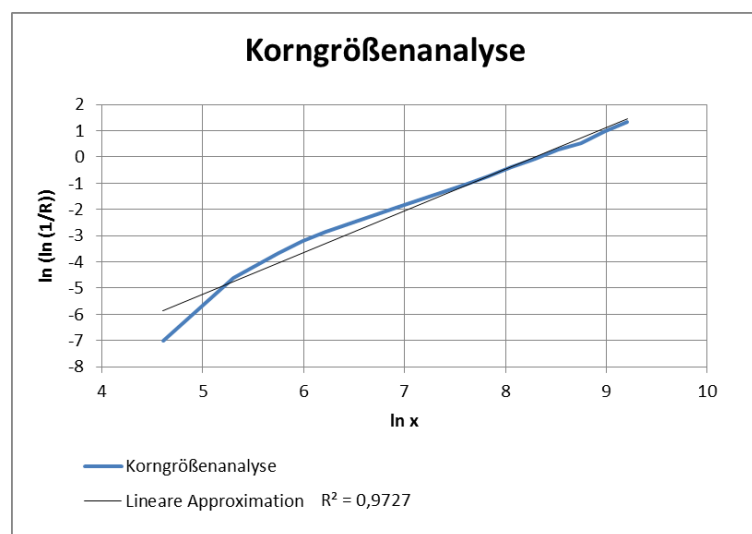
Partikelkollektive können durch die Wahl eines geeigneten Verteilungsgesetzes mittels weniger Parameter beschrieben werden und somit den weiteren Umgang (z.B. Berechnungen) erleichtern. Zu den wichtigsten Verteilungsfunktionen zählen die logarithmische Normalverteilung, Potenzfunktion (auch GGS-Funktion genannt) und die RRSB-Funktion. Bei der ersten Funktion ist die logarithmierte Korngröße normalverteilt und stellt eine gute Näherung bei Kollektiven mit einem hohen Feingutanteil dar. In der Regel wird sie zur Approximation von Tropfengrößenverteilungen durch Zerstäubungen verwendet. Die Potenzfunktion findet dagegen Anwendung in der Beschreibung von Korngrößenanalysen nach einer Grobzerkleinerung. Die RRSB-Verteilung stimmt besonders mit Partikelkollektiven überein, welche aus der Feinzerkleinerung stammen.

Alle Verteilungen können über zwei sogenannte Feinheitparameter beschrieben werden. Des Weiteren existiert für jeden der genannten Funktionen ein vorgefertigtes Wahrscheinlichkeitsnetz in dem die relative Durchgangssumme bzw. relative

Rückstandssumme, sowie die Korngröße der ermittelten Kornanalyse eingetragen werden können. Ergibt der Verlauf in einer der drei Netze eine Gerade, können diejenigen Feinheitparameter der entsprechenden Funktion für weitere analytische Auswertungen herangezogen werden. Im Falle der Grünmix-Verteilung zeigt sich eine lineare Abhängigkeit von $\ln(\ln(1/R))$ von $\ln x$ (siehe Tabelle 2), weswegen diese Kornanalyse der RRSB-Verteilung gehorcht. Der tiefste Punkt bei einer Korngröße von 0,1 mm zeigt eine deutliche Abweichung gegenüber den restlichen Werten. Dies ist sehr wahrscheinlich auf den geringen Messwert ($Q=0,09$ Vol.% bzw. $R=99,91$ Vol.%) der Randklasse der Verteilung zurückzuführen.

Tabelle 2: Links: Einfach logarithmierte Korngröße und doppelt logarithmische, inverse Rückstandssumme; Rechts: Darstellung der Korngrößenanalyse in einem doppelt logarithmischen Diagramm mit linearer Approximation

Logarithmierte Kornanalyse		
x [mm]	ln x	ln (ln (1/R))
10	9,21	1,33
8	8,99	0,99
6,3	8,75	0,54
5	8,52	0,29
4	8,29	-0,08
3,15	8,06	-0,40
2,5	7,82	-0,76
2	7,60	-1,06
1	6,91	-1,95
0,5	6,21	-2,85
0,4	5,99	-3,19
0,315	5,75	-3,67
0,2	5,30	-4,61
0,1	4,61	-7,01



Wie bereits erwähnt liegt bei einer RRSB-Verteilung eine lineare Proportionalität zwischen der doppelt logarithmierten, inversen Rückstandssumme und der einfach logarithmierten Korngröße vor. Als Funktion ausgedrückt lässt sich die Rückstandssumme bzw. Durchgangssumme gemäß einer RRSB-Funktion nach folgenden Formeln anschreiben.

$$R(x) = e^{\left(\frac{-x}{dp'}\right)^n}$$

Formel 2-1

$$Q(x) = 1 - e^{\left(\frac{-x}{dp'}\right)^n}$$

Formel 2-2

Durch das Auftragen der Durchgangssumme der Korngrößenverteilung in das RRSB-Netz, können die zwei Feinheitparameter abgelesen werden (siehe Anhang I). Dabei handelt es

sich zum einen um den Lageparameter dp' , jener Korngröße bei einer relativen Durchgangssumme von 63,2%. Dieser Wert stammt aus folgender Formel:

$$Q(dp') = 1 - e^{-1} = 63,2\%$$

Formel 2-3

Zum anderen wird die Breite der Verteilung durch den zweiten Feinheitsparameter angegeben, dem Gleichmäßigkeitsparameter n . Er kann ebenfalls einfach aus dem Diagramm abgelesen werden, indem die Gerade parallel durch den Achsenkreuzpunkt verschoben wird. Der Parameter wird am Randstab abgelesen und entspricht der Steigung der Verteilung. Neben der grafischen Methode können beide Werte auch berechnet werden, wofür die Kenntnis zweier Wertepaare (Q_i, x_i) aus der Korngrößenanalyse nötig ist.

$$n = \frac{\ln \ln \frac{1}{1-Q_1} - \ln \ln \frac{1}{1-Q_2}}{\ln \frac{x_1}{x_2}}$$

Formel 2-4

$$dp' = \frac{x_1}{\left(\ln \frac{1}{1-Q_1}\right)^{\frac{1}{n}}}$$

Formel 2-5

Da die Berechnung nur zwei Messpunkte miteinbezieht, ist hierbei besonders auf die Wahl dieser zu achten.

Die Approximation der Korngrößenverteilung mittels der RRSB-Funktion ermöglicht eine einfache Beschreibung der Analyse durch zwei Parameter. Dadurch sind mehrere Analysen auch untereinander schnell vergleichbar. Des Weiteren können daraus weitere Strukturparameter wie die spezifische Oberfläche und der Sauterdurchmesser ermittelt werden.

2.1.4 Spezifische Oberfläche und Sauterdurchmesser

Der Sauterdurchmesser d_{32} kann als Durchmesser einer monodispersen Schüttung beschrieben werden, dessen Gesamtvolumen- zu Oberflächenverhältnis jener des polydispersen Partikelkollektivs entspricht. Im Falle sphärischer Partikel kann der Sauterdurchmesser durch die Formel 2-6 [26] ermittelt werden.

$$d_{32} = \frac{6 * \text{gesamtes Volumen der Partikel}}{\text{gesamte Oberfläche der Partikel}} = \frac{6 * V_s}{A_s} = \frac{\sum n_i d_i^3}{\sum n_i d_i^2}$$

Formel 2-6

Für eine Verteilung gemäß RRSB kann der d_{32} auch über die Verteilungsparameter n und dp' mittels Gammafunktion Γ (siehe Formel 2-7 [26]) berechnet werden.

$$d_{32} = \frac{dp'}{\Gamma(1 - \frac{1}{n})}$$

Formel 2-7

Die volumenbezogene, spezifische Oberfläche $a_{V,S}$ ist definiert als der Quotient der gesamte Oberfläche aller Partikel durch das gesamte Volumen und ist somit umgekehrt proportional zum Sauterdurchmesser (siehe Formel 2-8). Des Weiteren existiert eine massebezogene, spezifische Oberfläche $a_{m,S}$, bei der sich die Partikeloberfläche nicht auf das Volumen, sondern auf die Partikelmasse bezieht. Beide Parameter hängen über die Feststoffdichte miteinander zusammen und können einfach nach Formel 2-9 berechnet werden.

$$a_{V,S} = \frac{A_s}{V_s} = \frac{6}{d_{32}}$$

Formel 2-8

$$a_{V,S} = \rho_s * a_{m,S}$$

Formel 2-9

2.1.5 Sphärizität und Formfaktor

Im Falle nicht sphärischer Partikel muss zusätzlich ein Faktor miteinberechnet werden, der die Abweichung von der Kugel bestimmt. Dafür stehen die Sphärizität Ψ , der Heywoodfaktor f und der Formfaktor ϕ zur Verfügung, welche verschiedene Strukturparameter eines realen Partikels verwenden, um diese mit einer Kugelform zu vergleichen. Die Sphärizität (im zweidimensionalen Fall auch Zirkularität genannt) ist dabei definiert über das Verhältnis der Oberflächen und nimmt für ein sphärisches Teilchen den Wert 1 an. Je stärker die Formabweichung, desto niedriger der Wert, da die Oberfläche eines stärker verformten Partikels größer wird.

$$\Psi = \frac{\text{Oberfläche einer volumengleichen Kugel}}{\text{tatsächliche Oberfläche}} ; \Psi \leq 1$$

Formel 2-10

Im Vergleich dazu bezieht sich der Heywoodfaktor f auf das Verhältnis der spezifischen Oberflächen.

$$f = \frac{\text{spez. Oberfläche des tatsächlichen Partikels}}{\text{spez. Oberfläche einer Kugel mit Durchmesser } x} = \frac{a_{V,S}}{6/x}$$

Formel 2-11

Anstelle der Durchmesservariable x kann nun ein Äquivalentdurchmesser eingesetzt werden, mit dessen Hilfe sich eine geometrische oder physikalische Eigenschaft eines unregelmäßig geformten Partikel auf eine Kugel- bzw. Kreisform übertragen lässt. Am häufigsten wird der Durchmesser einer volumengleichen Kugel d_v verwendet, weswegen der entstehenden Formel ein eigener Name gegeben wurde, der Formfaktor φ . Dieser ist außerdem der Kehrwert der Sphärizität und kann vereinfacht über den Durchmesser einer volumen- (d_v) und oberflächengleichen (d_s) Kugel ausgedrückt werden. Der Wertebereich beginnt bei $\varphi = 1$ für eine Kugel und nimmt mit stärkerer Verformung zu.

$$\varphi = \frac{1}{\Psi} = \left(\frac{d_s}{d_v}\right)^2 ; \varphi \geq 1$$

Formel 2-12

In der Berechnung der spezifischen Oberfläche von nicht sphärischen Partikel geht der Formfaktor in den Zähler der Formel ein.

$$a_{V,S} = \frac{6 * \varphi}{d_{32}}$$

Formel 2-13

Des Weiteren kann die spezifische Oberfläche aus dem RRSB-Netz herausgelesen werden. Die Vorgangsweise entspricht derselben wie bei der grafischen Bestimmung des Feinheitsparameters n , indem die Gerade wieder durch den Pol des Diagramms verschoben wird. Am Rand ist neben dem Gleichmäßigkeitsparameter meist ein zweiter Randstab für eine dimensionslose Oberflächenkennzahl K_s aufgetragen aus dem sich die spezifische Oberfläche berechnen lässt.

$$K_s = \frac{a_{V,S} * dp'}{\varphi}$$

Formel 2-14

2.1.6 Porosität

Ein weiterer wichtiger Parameter bei der Erstellung der Modellpackung ist die Porosität ε . Diese gibt den Hohlraumvolumenanteil in einer Schüttung bzw. eines porösen Feststoffes an und wirkt sich maßgeblich auf den Druckverlust und damit die Strömungsgeschwindigkeit bei der Durchströmung einer Schüttung aus.

$$\varepsilon = \frac{V_H}{V}$$

Formel 2-15

Grundsätzlich wird zwischen innerer (ε_i) und äußerer (ε_a) Porosität unterschieden. Erstere bezieht sich auf die Hohlräume innerhalb eines porösen Feststoffes, während die äußere Porosität jenes Lückenvolumen hervorgerufen durch geschüttete Körner beschreibt (siehe Abbildung 7). Die Gesamtporosität errechnet sich nach Formel 2-16.

$$\varepsilon_{ges} = \varepsilon_i + \varepsilon_a - \varepsilon_i * \varepsilon_a$$

Formel 2-16

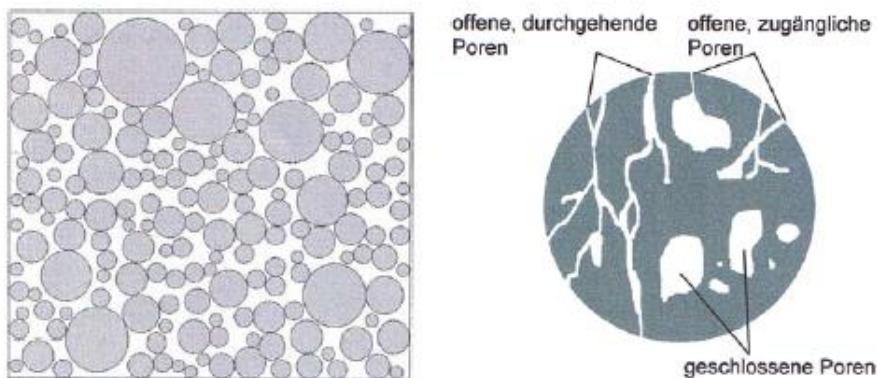


Abbildung 7: links: Schüttung aus sphärischen Partikeln; rechts: Darstellung eines porösen Korns [5]

Für die Strukturbildung werden vorerst nur kompakte Feststoffe verwendet, weswegen die innere Porosität wegfällt. Die Porosität stellt außerdem den Zusammenhang zwischen der Feststoffdichte und Schüttgutdichte her.

$$\rho_{schütt} = \rho_s * (1 - \varepsilon_{ges})$$

Formel 2-17

In der Praxis ist die Porosität von Schüttungen allerdings nur schwer bestimmbar. Einerseits kann sie bei unregelmäßig geformten Partikeln stark variieren, andererseits verändert sich bereits bei leichten Rütteln die Porosität. Am Sinterband kann der Lückenanteil der Schüttung einen Wertebereich zwischen 36-46% annehmen.

2.1.7 Approximierte Verteilung basierend auf den RRSB-Parametern

Aufgrund bereits genannter Vorteile einer Korngrößenverteilung gemäß einer Verteilungsfunktion wird anhand der RRSB-Feinheitsparameter eine neue, approximierte Korngrößenverteilung erstellt. Aus dem RRSB-Netz lassen sich folgende Daten gewinnen, wobei für den Formfaktor der inverse Wert der mittleren Sphärizität aus dem Analyseergebnis der photooptischen Messung verwendet wird. Das händisch ausgewertete Netz findet sich im Anhang I.

$$n = 1,4$$

$$dp' = 4 \text{ mm}$$

$$K_s = 16 \Rightarrow a_{V,S} = \frac{K_s * \varphi}{dp'} = \frac{16 * \left(\frac{1}{0,87}\right)}{4} = 4,60 \frac{\text{mm}^2}{\text{mm}^3}$$

Aus der analytischen Methode berechnen sich nachfolgende Werte nach Formel 2-4, 2-5, 2-7 und 2-13, wobei auch in diesem Fall auf denselben Formfaktor bzw. derselben Sphärizität zurückgegriffen wird. Die verwendeten Wertepaare (Q_i, x_i) werden dabei nicht willkürlich gewählt, da der Fehler unter Umständen enorm sein kann. Mittels der Software EXCEL wird deshalb eine Vielzahl von Feinheitsparametern durch die Kombination unterschiedlicher Wertepaare ermittelt und schließlich jenes Paar ausgewählt, welches den Mittelwert aller Wertepaare am nächsten liegt.

$$Q_1 = 93,14 \%$$

$$Q_2 = 2,51 \%$$

$$x_1 = 8 \text{ mm}$$

$$x_2 = 0,315 \text{ mm}$$

$$n = \frac{\ln \ln \frac{1}{1 - 0,9314} - \ln \ln \frac{1}{1 - 0,0251}}{\ln \frac{8}{0,315}} = 1,44$$

$$dp' = \frac{8}{\left[\ln \left(\frac{1}{1 - 0,9314}\right)\right]^{\frac{1}{1,44}}} = 4,03 \text{ mm}$$

$$d_{32} = \frac{4,03}{\Gamma\left(1 - \frac{1}{1,44}\right)} = 1,37 \text{ mm}$$

$$a_{V,S} = \frac{6 * \left(\frac{1}{0,87}\right)}{d_{32}} = 5,02 \frac{\text{mm}^2}{\text{mm}^3}$$

Tabelle 3 stellt die Werte für den Gleichmäßigkeitsparameter n , Lageparameter dp' , sowie die spezifische Oberfläche und den Sauterdurchmesser gegenüber, welche über grafische und analytische Methoden bestimmt wurden. Vergleichend wird ebenfalls die spezifische Oberfläche aus der photooptischen Analyse angegeben. Die Software des verwendeten Messgerätes ist in der Lage die gescannten Partikel einzeln abzuspeichern, ihren Umfang, sowie die Fläche zu messen und daraus Strukturparameter wie die spezifische Oberfläche (und mittlere Sphärizität) zu berechnen.

Tabelle 3: Ermittelte Werte für die RRSB-Verteilungsparameter, Sauterdurchmesser und spez. Oberfläche aus der Berechnung, grafischen Auswertung und der zur Verfügung gestellten Kornanalyse aus [5]

Vergleich von Parametern aus unterschiedlichen Analysemethoden				
Parameter		Berechnete Werte	Aus RRSB-Netz	Aus Kornanalyse
	Einheit			
n		1,44	1,4	
dp`	[mm]	4,03	4	
d ₃₂	[mm]	1,37		
a _{v,s}	[cm ² /cm ³]	50,16 ¹⁾	45,98 ¹⁾	35,61

1) Als Sphärizität wird der Wert $\Psi=0,87$ verwendet, welcher aus der photooptischen Kornanalyse stammt.

Für die approximierte Korngrößenverteilung werden schließlich die gerundeten, berechneten Werte verwendet und nach Formel 2-2 die neue Durchgangssumme ermittelt. Tabelle 4 zeigt diese Verteilung, welche nur mehr eine Nachkommastelle bezüglich der Korngröße aufweist. Sie besitzt 16 Kornklassen zwischen 0,1 und 12 mm, wobei jene Kornklasse zwischen 16-20 mm aufgrund ihres geringen Anteils ($\Delta Q = 0,06\%$) in der sphärischen Modellpackung weggelassen wird. Des Weiteren wird der relative Summenanteil jener Korngröße unter 0,1 mm mit der nächst höheren Kornklassen zusammengefasst. In GeoDict wird nur die untere Korngrenze (die in der Tabelle 4 angegebene Korngröße) mit ihrem jeweiligen relativen Summenanteil (gerundet auf zwei Nachkommastellen) eingegeben.

Tabelle 4: Approximierte Verteilung mittels der errechneten RRSB-Parameter

n	1,44
dp`	4

x [mm]	Q [%]	ΔQ [%]	ΔQ [V%] alt
20	100,00		0
16	99,94	0,06	0
12	99,23	0,71	0
10	97,63	1,60	2,29
8	93,37	4,26	4,57
6	83,35	10,01	11,12
5	74,82	8,54	8,38
4	63,21	11,60	13,5
3	48,36	14,85	11,31
2	30,83	17,53	19,63
1	12,70	18,13	15,97
0,5	4,88	7,82	7,61
0,4	3,57	1,32	1,58
0,3	2,37	1,20	1,53
0,2	1,33	1,04	1,52
0,1	0,49	0,84	0,9
<0,1		0,49	0,09

2.1.8 Präsentation der erstellten sphärischen Packung

Tabelle 5: Parameter und Eigenschaften der erstellten Kugelpackung

Parameter der erstellten Kugelpackung		
NXxNYxNZ [Anzahl]		300x300x500
Voxel [μm]	49,7	
Material		Glas
	Poren	Luft
Porosität	0,4594	
Anzahl Kugeln	70	
Quader [mm]	Fläche	14,9214x14,9214
	Höhe	24,8695

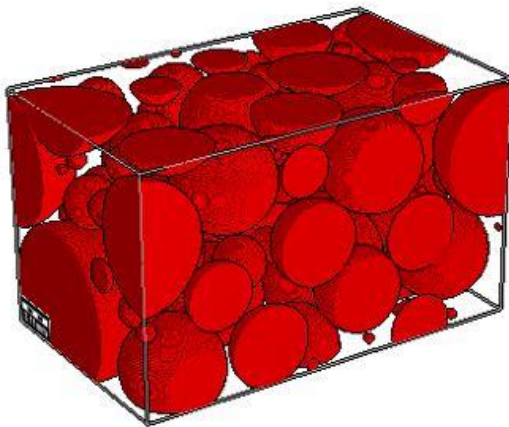


Abbildung 8: 3D-Modell der fertigen Kugelpackung mit einer Korngrößenverteilung gemäß RRSB-Parameter ($n=1,44$, $dp'=4$) in einer Box mit den Abmaßen von 14,92x14,92 mm und einer Voxelgröße von 49,7 μm

Die erzeugte Struktur besitzt eine Porosität von rund 45,94% und eine Voxelgröße von 49,7 μm . Bei einer Anzahl von 300x300x500 Voxel ergibt dies eine Packung mit einer quadratischen Grundfläche und einer Seitenlänge von 14,92 mm. In der Höhe misst der Quader 24,87 mm. Die Abmaße ergeben sich durch eine Reihe von Testläufen. Die Packung darf zum einen nicht zu klein sein, da sie die Korngrößenverteilung und vorgegebene Porosität möglichst repräsentativ wiedergeben soll. Zum anderen führt eine zu große Box zu einer stark erhöhten Rechenleistung und somit auch Rechenzeit. Die Voxelgröße kann in diesem Fall nicht eingegeben werden, sondern wird automatisch erstellt. Obwohl keine genauen Angaben in der Beschreibung enthalten sind, ist diese vermutlich abhängig von der Korngröße, Anzahl der Kugeln und der Zielporosität. Die 70 Kugeln sind in einer periodischen Struktur verteilt, was bedeutet, wenn ein Partikel sich mit der Randfläche überschneidet, wird der Körper auf der gegenüberliegenden Seite fortgeführt und nicht abgeschnitten. Als Material für die Partikel wird Glas gewählt und der Porenraum ist mit Luft gefüllt. Für die Durchströmungssimulationen sind die physikalischen Materialeigenschaften der Körner, wie die Dichte, Wärmeleitfähigkeit, etc. nicht relevant, weswegen als exemplarisches Material Glas gewählt wurde. Die

geometrischen Eigenschaften der sphärischen Modellpackung sind in Tabelle 5 zusammengefasst.

2.2 Mehreckige Modellpackung mit Korngrößenverteilung gemäß RRSB

In der zweiten Modellpackung wird die Morphologie der realen Sinterkörner durch Änderung der Oberfläche und Form besser angenähert. Die Partikel in der Packung sind als Polyeder aufgebaut, welche in einer sphärischen und ellipsoiden Grundform eingebettet sind. Die Größe der Körner folgt derselben approximierten Verteilung gemäß der ermittelten RRSB-Parameter wie bei der Kugelpackung (siehe Tabelle 4). Auch dieselbe Porosität ($\varepsilon = 0,46$) wird hierbei angenähert. Allerdings muss aufgrund der Abweichung von der Kugelform der Formfaktor für weitere Berechnungen berücksichtigt werden, besonders bei der Ermittlung der spezifischen Oberfläche. Der Packungsbereich wird wieder periodisch modelliert. Im nachfolgenden wird die Erstellung dieser Modellpackung, ihre Eigenschaften, sowie die Problematik bei der Generierung erläutert.

2.2.1 Generierung der Kornformgruppen

Die 14 Kornklassen der approximierten Verteilung werden für die Generierung in vier Gruppen aufgeteilt: Groß, Groß-mittel, Klein-mittel und Klein. Jede Gruppe besitzt dabei eine eigene Form und Farbe bzw. Material ID, um unerwünschte Veränderung in der Kornstruktur, z.B. plötzliches Überlappen, Verzerrungen oder einen stark abweichenden Feststoffanteil einer bestimmten Korngruppe beobachten zu können. Tabelle 6 zeigt übersichtlich die vier erzeugten Kornformgruppen, ihre Klassenbreite, sowie die jeweiligen Volumenanteile. Bei diesem Modell wird im Gegensatz zur Kugelpackung auch die größte Kornklasse durch ihre untere Korngrenze ($x = 16 \text{ mm}$) berücksichtigt.

Die Software erlaubt neben Kugeln die Erzeugung polyedrischer Strukturen mit einer definierten Anzahl an Eckpunkten. Dem Partikel wird hierbei eine Begrenzung durch eine umschließende Figur vorgeben. Die Ecken werden willkürlich auf der Hüllfläche dieser geometrischen Figur gesetzt und durch gerade Kanten verbunden. Je mehr Ecken vorgegeben werden, desto mehr nähert sich das erzeugte Partikel der umschließenden Figur an. Je nach Kantigkeit kann somit das Aussehen mehrerer Körner für ein und dieselbe Kornklasse variieren und nähert sich damit der realen Struktur des Rohmix-Materiales an.

Tabelle 6: Klassenbreiten der gewählten Korngrößenklassen und Volumenanteile der jeweiligen Kornformgruppe

Kornformgruppen mit Anteile		
	[mm]	ΔQ [vol.%]
Groß	8-16	6,6
Groß-mittel	4-6	30,2
Klein-mittel	1-3	50,5
Klein	0,1-0,5	12,7

Gruppe Groß

In der größten Gruppe befinden sich alle Kornklassen mit einem Durchmesser zwischen 8 und 16 mm. Ihnen wird als begrenzende Grundform ein Ellipsoid mit 10 Eckpunkten zugewiesen, wodurch die Partikel eine starke Kantigkeit zeigen. Um eine dreidimensionale Ellipsenform zu erzeugen, müssen, wie in Abbildung 9 dargestellt, drei Durchmesser angegeben werden. Die Werte für den Durchmesser 1 stammen dabei von der approximierten Kornanalyse. Die restlichen Durchmesser werden anhand realer Partikel aus einer Grünmix-Schüttung ermittelt, welche stichprobenartig aus einer Schüttung entnommen und händisch gemessen werden. Für jede der drei Durchmesser wird ein Mindest- und Maximalwert definiert. Innerhalb der angegebenen Grenzen kann die Verteilung als gleichverteilt angenommen werden, da diese Gruppe den kleinsten Anteil besitzt (~ 6 %). Des Weiteren wird allen Körnern eine isotrope Ausrichtung zugeschrieben, das heißt dass diese keine bestimmte Richtung bezüglich ihrer Achsenorientierung in der Packung besitzen, sondern zufällig platziert werden. In Tabelle 7 sind die wichtigsten Eckdaten für die Kornformgruppe Groß zusammengefasst.

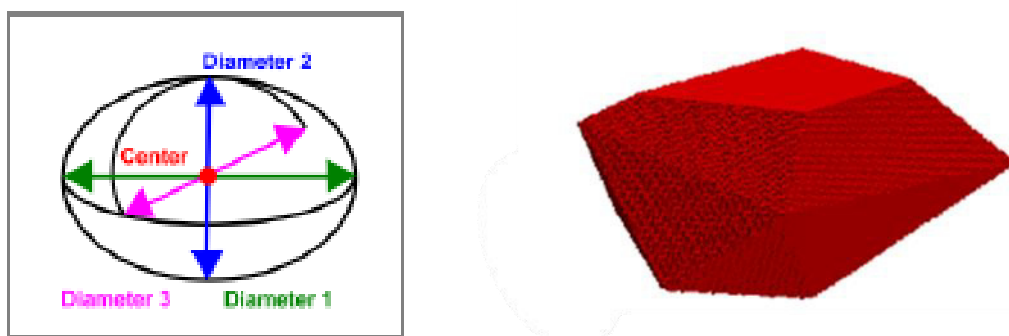


Abbildung 9: Links: Ellipsoid mit Darstellung der Achsendurchmesser; Rechts: ein exemplarisches Korn aus der Gruppe Groß

Tabelle 7: Übersicht über alle Parameter der Kornformgruppe Groß

Gruppe Groß		
Grundform	Ellipsoid	
Anzahl der Eckpunkte	10	
Durchmesser 1 [mm]	Min	8
	Max	16
Durchmesser 2 [mm]	Min	5
	Max	15
Durchmesser 3 [mm]	Min	2
	Max	10
Anzahl der Kornklassen	4	
Kornanteil ΔQ [V%]	6,6	
Verteilung	gleichverteilt	
Ausrichtung	isotrop	

Gruppe Groß-mittel

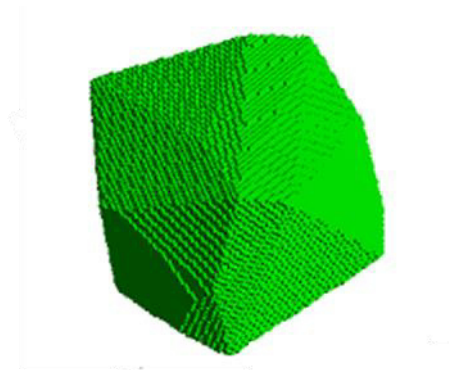


Abbildung 10: Exemplarisches Korn aus der Kornformgruppe Groß-mittel

In der zweiten Gruppe (Groß-mittel) sind drei Kornklassen zusammengefasst, welche eine Größe zwischen 4-6 mm besitzen. Ihre Partikel besitzen eine sphärische Grundstruktur, an deren Grenzfläche 20 Eckpunkte gesetzt werden und sich somit der Kugelform mehr annähern. Abbildung 10 illustriert exemplarisch ein einzelnes Korn aus der Gruppe, welches aufgrund der hohen Eckzahl große Ähnlichkeit mit den restlichen Körnern derselben Fraktion besitzt. Auch hier wird ein Minimal- und Maximaldurchmesser benötigt und eine Gleichverteilung innerhalb dieses Intervalls angenommen, da die zusammengefassten Kornklassen ungefähr denselben Anteil ($\Delta Q \sim 10$ Vol.%) besitzen. Tabelle 8 fasst nochmals die wichtigsten Daten zusammen.

Tabelle 8: Übersicht über alle Parameter der Kornformgruppe Groß-mittel

Gruppe Groß-mittel		
Grundform	Kugel	
Anzahl der Eckpunkte	20	
Durchmesser [mm]	Min	4
	Max	6
Anzahl der Kornklassen	3	
Kornanteil ΔQ [%]	30,2	
Verteilung	gleichverteilt	
Ausrichtung	isotrop	

Gruppe Klein-mittel

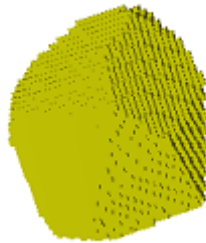


Abbildung 11: Exemplarisches Partikel aus der Kornformgruppe Klein-mittel

In der Klein-mittel Gruppe sind wieder drei Kornklassen zwischen 1-3 mm enthalten, für welche allerdings keine Gleichverteilung mehr angenähert wird. Den unteren Korngrenzen der Klassen werden stattdessen direkt die jeweiligen Anteile zugeordnet, welche in Tabelle 9 aufgelistet sind. Da bei dieser Art der Eingabe in die Software die Summe der Anteile 100 % ergeben muss, sind die angegebenen Werte höher als in Tabelle 4. Das Verhältnis bleibt allerdings gleich. Diese Gruppe besitzt mit einem Gesamtanteil von 50,5 % den größten Anteil. Für die Grundform wird eine polyedrische Kugel mit 30 Eckpunkten vorgegeben (siehe Abbildung 11).

Tabelle 9: Übersicht über alle Parameter der Kornformgruppe Klein-mittel

Gruppe Klein-mittel		
Grundform	Kugel	
Anzahl der Eckpunkte	30	
Kornklassen in [mm] mit Anteil in [%]	3	29
	2	35
	1	36
Kornanteil gesamt ΔQ [%]	50,5	
Ausrichtung	isotrop	

Gruppe Klein

Die letzte Kornformgruppe umfasst alle Kornklassen unter 1 mm, für welche vereinfacht eine perfekte Kugelform angenommen wird, da eine polyedrische Struktur aufgrund der geringen Größe vernachlässigt werden kann. Für die fünf Kornklassen werden die Anteile wieder direkt pro Kornklasse eingegeben, wobei auch hier die Summer der jeweiligen Kornklassenanteile 100% ergeben muss.

Tabelle 10: Übersicht über alle Parameter der Kornformgruppe Klein

Gruppe Klein		
Grundform	Kugel	
Kornklassen in [mm] mit Anteil in [%]	0,5	61,6
	0,4	10,4
	0,3	9,3
	0,2	8,2
	0,1	10,5
Kornanteil gesamt ΔQ [V%]	12,7	
Ausrichtung	isotrop	

2.2.2 Ansätze zur Erstellung einer dichten Packung

Eine Generierung der kompletten Struktur in einem Schritt führt zu einem Absturz der Software, obwohl die Packung auf derselben Korngrößenverteilung und derselben Porosität wie das sphärische Modell basiert. Neben der hohen Packungsdichte spielt die komplexe Kornform eine große Rolle, was durch das zusätzliche Überlappungsverbot erschwert wird. Diese Unterfunktion der Software, mit welcher ein Überlappen verhindert werden kann, ist laut Hersteller nur für geringe Packungsdichten oder einfache sphärische Formen geeignet.

Um ein Abstürzen zu verhindern werden verschiedene Lösungsansätze getestet, welche in den nachfolgenden Unterkapiteln detaillierter erklärt werden. Die fertige Packung ist eine Kombination dieser Versuche und wird schließlich in Kapitel 2.3.3 vorgestellt. Zu den Ansätzen zählen:

- Lösungsansatz I: „Schrittweises“ Einfügen der Gruppen: Partikel werden nacheinander in ein fertiges Gebiet platziert
- Lösungsansatz II: „seed & grow“: Platzieren von kleineren Körnern und wachsen lassen
- Lösungsansatz III: „Upscaled Domain“: Packungsgebiet nach Partikelanordnung komprimieren

Lösungsansatz I: „Schrittweises“ Einfügen der Gruppen

Nachdem die Kornklassen in vier Kornformgruppen aufgeteilt sind, werden im ersten Versuch diese nicht in einem Schritt, sondern nacheinander eingefügt. Angefangen wird mit der größten Gruppe, da ein nachträgliches Einfügen der großen Körner in einer Schüttung aus vielen kleinen, regelmäßig verteilten Partikeln unmöglich ist. Ist ein Korn einmal platziert kann es nicht mehr von seiner Position verschoben werden. Mit der ersten Gruppe wird auch die Größe des Packungsbereichs festgelegt und alle weiteren Gruppen in diesen hinzugefügt. Allerdings treten schon bei dem Versuch die dritte Gruppe einzuordnen die zuvor angeführten Probleme (Abstürzen der Software) auf, weswegen weitere Alternativen erkundet werden müssen.

Lösungsansatz II: „seed & grow“

In einer weiteren Idee wird versucht die Kornformgruppen zwar nacheinander, aber mit kleinerem Durchmesser einzufügen und nachträglich auf Originalgröße zu wachsen. GeoDict bietet dahingehend zwei Möglichkeiten an: „Grow Sediment“ und „Sinter“.

„seed & grow“: „Grow Sediment“

Die Funktion „Grow Sediment“ lässt die Körner durch das Auftragen von Schichten auf deren Oberfläche wachsen. In Abbildung 12 sind zwei Ausschnitte vor (links) und nach (rechts) dem Anwenden der Funktion „Grow Sediment“ dargestellt. Die Generierung des Ausgangsmodells (Abbildung 10: links) erfolgt nach Lösungsansatz I, allerdings werden die Eingabewerte für die Abmaße der Partikeldurchmesser halbiert und gleichzeitig auch das Feststoffvolumen auf seine Hälfte reduziert, womit die Porosität 73 % beträgt. Der Kornanteil muss daher für jede Kornformgruppe dementsprechend verkleinert werden. Dafür wird der neue Feststoffanteil einfach mit dem alten Kornanteil multipliziert. Tabelle 11 listet die neuen Anteile der Kornformgruppen auf, die sich aufgrund der erhöhten Porosität ergeben.

Tabelle 11: Neue Anteile der Kornformgruppen bei einer Porosität von 73 %

Feststoffanteil [%]	27
Porosität [%]	73

Kornformgruppe	Anteil [%]
Groß	1,78
Gr.-mittel	8,15
Kl.-mittel	13,64
Klein	3,43

Bei der Anwendung von „Grow Sediment“ wird dann das gewünschte Feststoffvolumen von 54 % eingegeben und die Oberfläche der Körner zusätzlich mit einer blauen Deckschicht belegt. Diese nimmt in dem vorliegenden Endergebnis rund 13 % des Raumes ein, während der restliche rote Anteil bei ca. 41 % liegt. Die Ecken und Kanten der Partikel werden in Folge der Vergrößerung abgerundet, wodurch die gewünschte Kantigkeit verloren geht. Des

Weiteren scheinen die Partikel proportional zu ihrer Krümmung zu wachsen, da die kleinen Körner stark anwachsen, während die größeren Partikeln nur ein geringes Wachstum erfahren. Der Vorteil der Funktion liegt in im geringen Rechenaufwand, sodass die Berechnungen schnell abgeschlossen sind. Allerdings überlappen zum einen die Partikel punktuell und zum anderen ist aufgrund des unterschiedlichen Wachstums die vorgesehene Korngrößenverteilung nicht mehr gegeben.

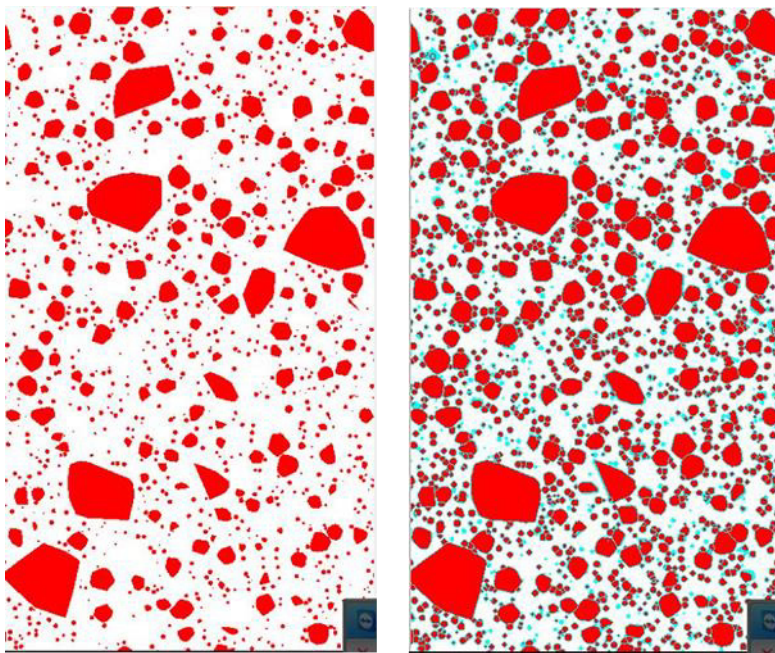


Abbildung 12: Links: Schnitt durch die Packung in der Z-Y-Ebene vor dem Anwachsen mit „Grow Sediment“; Rechts: derselbe Schnitt durch die Packung nach dem Anwachsen

„seed & grow“: „Sinter“

Im Gegensatz zur Schichttechnik werden bei der Funktion „Sinter“ die Partikel aufskaliert, wodurch diese ihre charakteristischen Kanten nicht verlieren. Die Generierung der Grundstruktur (siehe Abbildung 11 links) erfolgt auf dieselbe Art und Weise wie schon bei „Grow Sediment“, das heißt die dargestellte Packung besteht aus auf die Hälfte geschrumpfte Partikeln mit nur einem halb so hohen Feststoffvolumen. Zum besseren Erkennen der unterschiedlichen Kornformgruppen werden diese mit unterschiedlichen Farben dargestellt: Rot stellt dabei die größte Gruppe dar, Hellblau die Gruppe „Groß-mittel“, Gelb die Gruppe „Klein-mittel“ und Dunkelblau gibt schließlich die kleinste Gruppe wieder. Die Oberfläche ist nach dem Anwachsen wieder mit einer Schicht anders farbigen Materials überdeckt. Ziel ist es, abermals ein Feststoffvolumen von mindestens 54 % zu erreichen. Allerdings sind im Vergleich zur vorhergehenden Funktion starke Überlappungen zu sehen (siehe Abbildung 11 rechts), wodurch die vorgegebene Korngrößenverteilung nicht mehr erreicht wird.

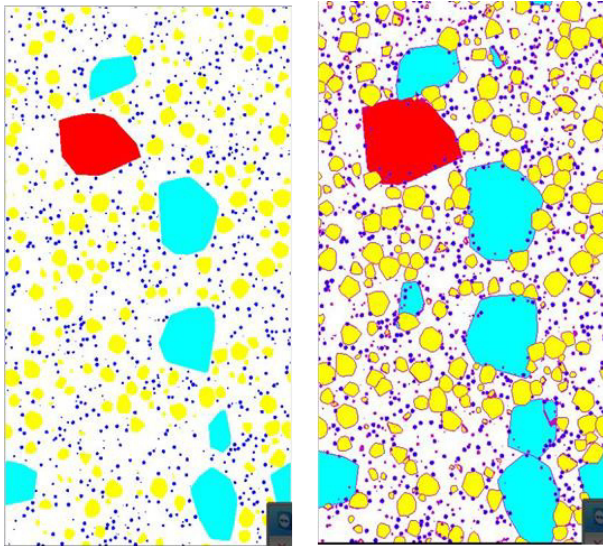


Abbildung 13: Links: Schnitt durch die Packung in Z-Y-Ebene vor dem „Sintern“; Rechts: derselbe Schnitt durch die Packung nach dem Sintern

Vergleich und Ergebnisse beider Funktionen

Im Vergleich beider Funktionen sind trotz desselben Ziels, nämlich ein Wachsen der Körner, starke Unterschiede zu sehen. Die Partikel nach dem Anwenden von „Sinter“ wachsen viel stärker, obwohl der Feststoffanteil in beiden Fällen gleich ist. Der Grund liegt im Verhältnis zwischen Volumen und Oberfläche, welches sich bei „Grow Sediment“ durch die Abrundung der Kanten ändert. Abbildung 14 verdeutlicht noch einmal in einem exemplarischen Beispiel den Unterschied beim Anwenden der zwei Funktionen auf eine nicht sphärische Struktur. Allerdings führen beide Varianten der Kornvergrößerung zu keiner direkt verwertbaren Packungen, da sich die Korngrößenverteilung in beiden Fällen signifikant ändert. Bei „Grow Sediment“ wachsen zwar die Körner nicht sehr stark ineinander, aber der Wachstumsalgorithmus führt zu einer unregelmäßigen Feststoffzunahme. „Sinter“ hingegen bietet zwar das gewünschte Wachstum, indessen Folge aber starke Überlappungen entstehen. Aufgrund dieser Probleme werden diese Methoden für die Packungsgenerierung nicht weiterverfolgt.

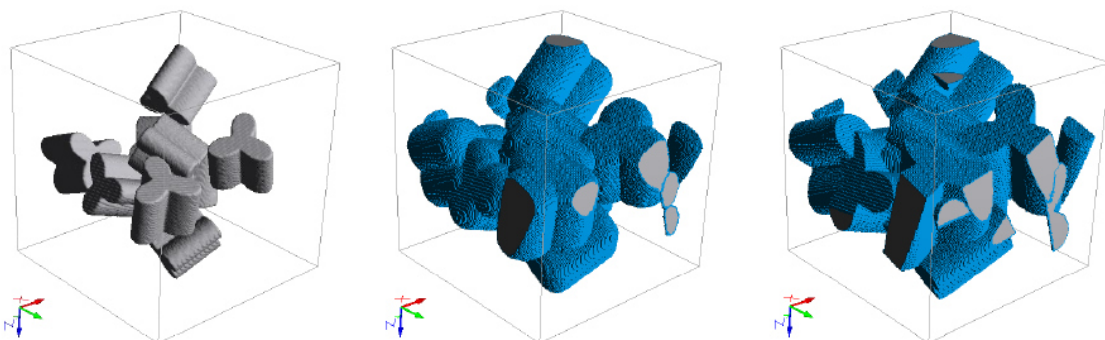


Abbildung 14: Vergleich der beiden Funktionen „Grow Sediment“ (mitte) und „Sinter“ (rechts) für nicht-sphärische Strukturen [12]

Lösungsansatz III: „Upscaled Domain“

Eine andere Herangehensweise umfasst das Einfügen der Korngrößenverteilung in ein größeres Gebiet, welches im Nachhinein komprimiert werden soll. Die in Kornformgruppen aufgegliederten Kornklassen werden wieder nach dem Prinzip des ersten Lösungsansatzes („Schrittweises“ Einfügen der Gruppen) in ein Gebiet mit definierten Abmaßen eingefügt. Die Korngrößen in der Verteilung werden dieses Mal nach dem im Kapitel 2.2.1 angeführten Werten beibehalten. Im Gegenzug ist das Gebiet in seinen Dimensionen größer und schafft damit den Partikeln mehr Raum zum Platzieren.

Zuerst wird nur die Höhe bzw. die Z-Richtung der Modellbox für das Einfügen verdoppelt, womit sich auch der Feststoffanteil wie schon im zweiten Lösungsansatz halbiert. Die Porosität der beträgt somit 73 %. Der Kornanteil jeder Kornformgruppe muss daher dementsprechend nach Tabelle 11 geändert werden. Nach erfolgreichem Einfügen wird die Höhe der Packung auf die Hälfte komprimiert, wobei der Vergleich vor und nach dem Quetschen in Z-Richtung in Abbildung 15 dargestellt ist. Die Farbcodierung der Kornformgruppen ist ähnlich jener Packung von „Sinter“ aus dem zweiten Lösungsansatz, allerdings ist hierbei das zweitgrößte Kollektiv nicht Hellblau, sondern Grün. Die Porosität und der Volumenanteil der einzelnen Gruppen des fertigen Modells entsprechen zwar den Vorgaben, aber das entstandene Gebilde zeigt starke Verzerrungen in Z-Richtung.

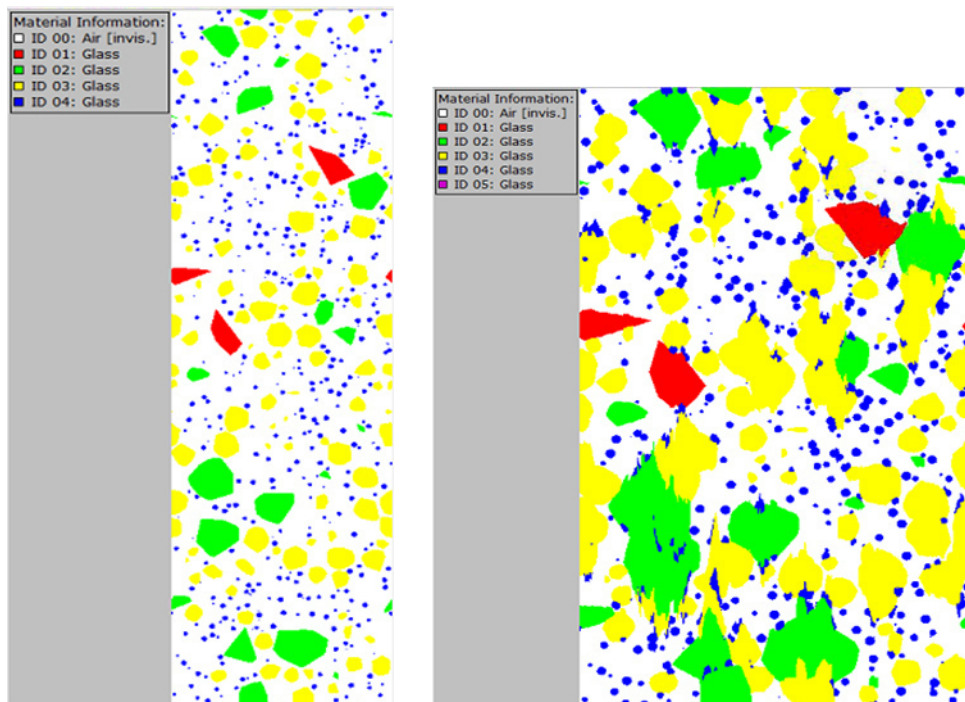


Abbildung 15: Links: Schnitt durch die Packung in der Z-Y-Ebene vor dem Komprimieren in Z-Richtung; Rechts: derselbe Schnitt durch die Packung nach dem Komprimieren

Um diese Verzerrungseffekte zu mindern werden nun die Abmaße der Ausgangsbox in jede Richtung vergrößert und schlussendlich das Modell auch von allen Seiten komprimiert. Die Packungsgröße wird dabei so gewählt, dass sich das Feststoffvolumen abermals auf die Hälfte reduziert. Die Voxelanzahl des Ausgangsmodells beträgt $500 \times 500 \times 896$ Voxel, während das neue Gebiet nach der Kompression nur mehr $400 \times 400 \times 700$ Voxel beträgt. Die quadratische Grundfläche wird also beibehalten und ihre Seiten um 20 % (bzw. $\frac{1}{1,25}$) verkleinert. Der unrunde Wert für die Ausgangshöhe mit 896 Voxel ergibt sich rechnerisch durch die zu erfüllenden Vorgaben. Diese sind zum einen ein ähnlicher Kompressionsfaktor für jede Seite und andererseits die Halbierung des Packungsvolumens zum Erzielen einer gewünschten Porosität von rund 46 %. Daraus heraus ergibt sich ein Faktor von 0,21875, welcher in GeoDict zur Kompression eingegeben werden muss. Das Ergebnis ist in Abbildung 16 auf der rechten Seite darstellt. Dieses Mal sind nur geringe Verzerrungen zu sehen und die Oberfläche erscheint etwas rauer. Porosität und Volumenanteile der Kornklassen entsprechen den Vorgaben und es existieren viele Kontaktflächen zwischen den Partikeln. Allerdings sind auch geringfügige Überlappungen zu erkennen. Diese Vorgehensweise liefert das beste Ergebnis und wird in weiterer Folge für spätere numerische Strömungssimulationen verwendet. Die genauen Eckdaten des fertigen Modells sind nachfolgend genauer erläutert.

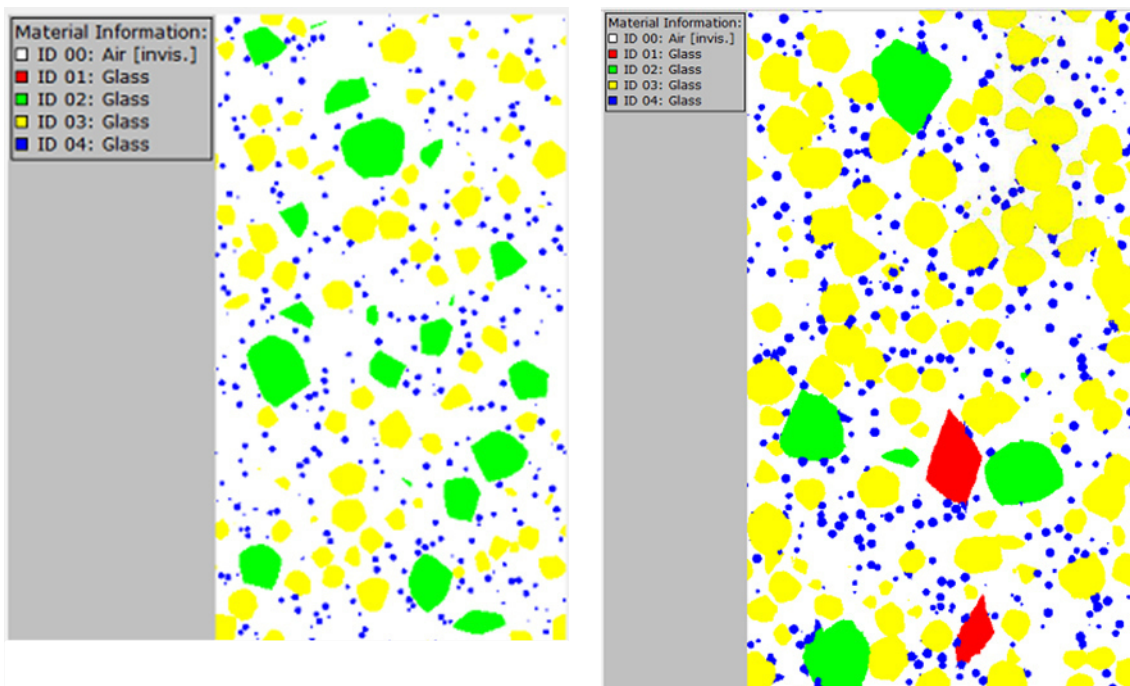


Abbildung 16: Links: Schnitt durch die Packung in der Z-Y-Ebene vor der allseitigen Komprimierung; Rechts: derselbe Schnitt durch die Packung nach der allseitigen Komprimierung

2.2.3 Präsentation der erstellten polyedrischen Packung

Die fertige Modellpackung besitzt bei einer Voxelzahl von 400x400x700 und einer Voxelgröße von 50 µm Abmaße mit 20x20x35 mm und ist damit größer als das sphärische Modell (siehe Tabelle 5). Durch die einfache Modellierung gemäß schrittweisem Gruppen-Einfügen (S.24, Lösungsansatz I) in Kombination mit Domain-Verzerrung (S.27, Lösungsansatz III) dauert die Rechenzeit nur einige Stunden und die Dimensionen können ohne großen Aufwand erhöht werden. Dadurch werden Korngrößenverteilung und Packungsdichte noch besser wiedergegeben. Für die spätere Durchströmung wird insbesondere die Höhe bzw. die Z-Richtung des Gebietes verlängert, um den Druckverlust der Schüttung besser analysieren zu können. Der Wert der Voxelgröße kann bei dieser Art der Generierung im Gegensatz zum sphärischen Modell selbst eingegeben werden, wird aber aufgrund der ausreichenden Auflösung einfach in aufgerundeter Form von der Kugelpackung übernommen. Als Kornmaterial wird wieder Glas hinterlegt und die Poren sind mit Luft gefüllt. Die Korngrößen der Partikel sind grundsätzlich der approximierten Korngrößenverteilung gemäß der ermittelten RRSB-Parameter aus Tabelle 4 entnommen. Allerdings ergeben sich bei der Generierung geringfügige Abweichungen, sodass die Porosität mit 46,13 % leicht erhöht ist gegenüber der Soll-Porosität von 46 %. In Tabelle 12 sind neben den restlichen Daten (z.B. Packungsabmaße, Voxelgröße, etc.) auch die erreichten Kornanteile der Kornformgruppen, sowie zum Vergleich ihre Sollwerte aufgelistet. Diese aufsummiert ergibt den Feststoffanteil. Abbildung 17 stellt ein dreidimensionales Bild der erstellten Packung dar, worin die vier Kornformgruppen durch ihre unterschiedliche Farbcodierung gut zu erkennen sind. Das Modell wirkt im Vergleich zur sphärischen Packung viel kompakter und es wird bei der späteren Strömungssimulation auch ein höherer Druckverlust erwartet.

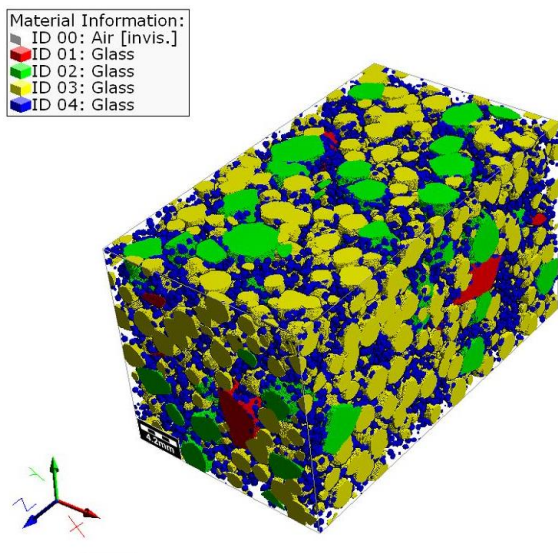


Abbildung 17: 3D-Darstellung der fertigen Packung aus polyedrischen Formen mit einer Korngrößenverteilung gemäß RRSB-Parametern ($n = 1,44$, $dp' = 4$) in einer Box mit den Abmaßen von 20x20x35 mm und einer Voxelgröße 50 µm

Tabelle 12: Parameter und Eigenschaften der erstellten Modellpackung aus polyedrischen Formen

Packungsparameter			
NXxNYxNZ [Anzahl]	400x400x700		
Voxelgröße	50	[µm]	
Quader [mm]	Fläche	20x20	
	Höhe	35	
Material	Glas		
	Poren	Luft	
	Ist	Soll	
Porosität	46,13	46	[%]
Groß	3,42	3,56	[%]
Mittel-groß	16,30	16,31	[%]
Mittel-klein	27,29	27,27	[%]
Klein	6,86	6,86	[%]
Summe	53,87	54	[%]

2.3 Freie Positionierung von skalierten Polyedern

Die vorhergehende Packung zeigt gegenüber der sphärischen Packung bereits erhebliche Verbesserungen bezüglich der Kornform, allerdings ist die Grundform der Einzelpartikel trotzdem meist (annähernd) kugelförmig. Genauere Betrachtungen des verwendeten Grünmix-Materials zeigen eine weite Bandbreite an Formen, welche von stabförmig, über abgeflacht zu tetraedrisch reichen. In einem neuen Ansatz werden Modelle einzelner Körner entwickelt, welche alle diese unterschiedlichen Strukturen besitzen und die Formvielfalt besser wiedergeben sollen.

2.3.1 Neue Korngrößenverteilung

Die Korngrößenverteilung soll für diese Modellpackung auf neuen Grünmix-Proben basieren, welche wiederum von externen Personen zur Verfügung gestellt wird. Im Zuge ihrer Bachelorarbeit analysierten Voller und Lindthaler händisch drei Mischgutproben von Grünbettschüttungen per Sieb [25], deren relative Durchgänge vergleichend in Abbildung 18 illustriert sind. Auffällig ist dabei der unterschiedliche Feinanteil (<1 mm) zwischen den Proben. Die starke Differenz ist auf die unterschiedliche Lagerzeit zwischen der Probeentnahme und Analyse zurückzuführen, denn je länger diese ist, desto trockener wird das Gut. Wie bereits am Anfang des zweiten Kapitels auf Seite 6 erwähnt, kleben die kleinsten Partikel durch die Feuchte an den größeren Körnern. Für die Lagerung wird das Mischgut in geschlossene Plastikkübeln bei Raumtemperatur gegeben, worin das Wasser zu kondensieren beginnt und sich am Deckel und den Wänden sammelt. Wird nun das Kollektiv gesiebt, ist die Haftkraft für die kleinen Körner nicht mehr gegeben und der Feinanteil steigt.

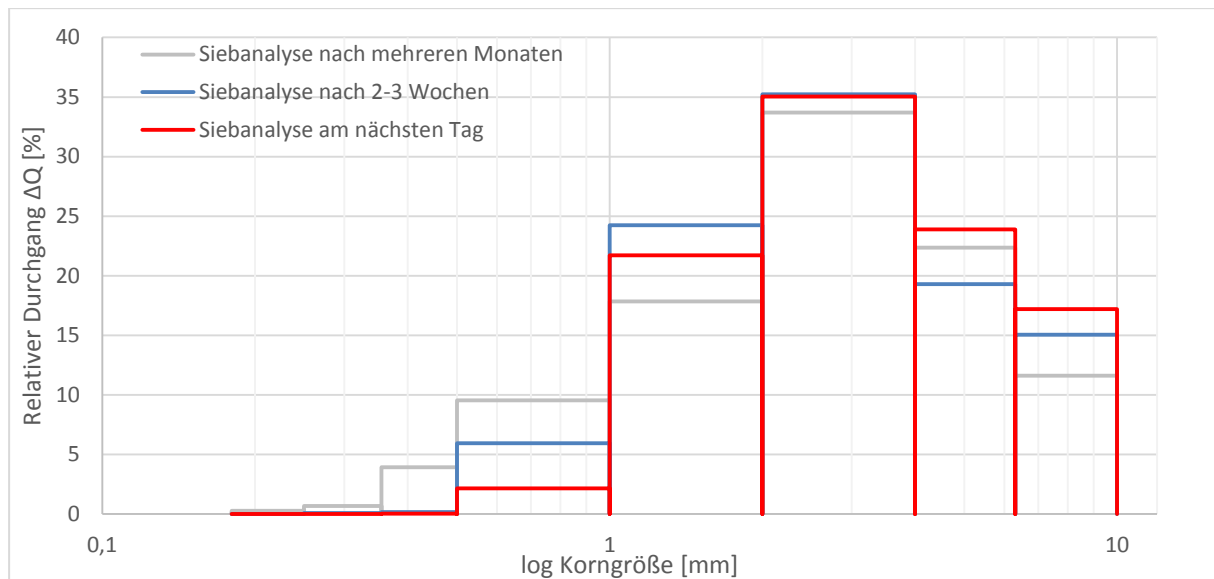


Abbildung 18: Relativer Durchgang von unterschiedlich lange gelagerten Proben eines Sinterrohmmixes; Daten aus [25]

Eine Nachahmung des ursprünglichen Zustandes ist nur sehr schwer möglich, da selbst durch die nachträgliche Zugabe von Wasser der Rolliereffekt fehlt. Des Weiteren ist bei einer längeren Lagerzeit der noch verbleibende Wasseranteil im Gut unbekannt. Deswegen wird empfohlen neben der Repräsentativität auch auf die Lagerzeit zu achten und eine Analyse bezüglich der Korngrößenverteilung so schnell wie möglich nach der Probeentnahme durchzuführen. Eine „falsche“ Verteilung würde das Modell und auch die nachfolgende Simulation verfälschen.

Aufgrund der händischen Durchführung der Siebanalyse ist die Anzahl und Auswahl der Kornklassen stark beschränkt. Im Gegensatz zur zuvor verwendeten Korngrößenanalyse aus [5] (siehe Tabelle 1), bei der die größte untere Kornklassengrenze theoretisch bis 25 mm ginge, liegt hier die größte untere Kornklassengrenze bei 6,3 mm. Aufgrund mangelnder Informationen wird eine formale obere Grenze von 10 mm eingeführt. In seiner Anzahl ist die Siebanalyse auf 8 Kornklassen beschränkt (zum Vergleich: aus der photooptischen Analyse gehen 14 Kornklassen hervor). Aus den drei zur Verfügung gestellten Proben fällt die Entscheidung eindeutig auf das in Abbildung 18 dargestellte rote Histogramm, welche bereits am nächsten Tag der Entnahme analysiert wurde. Eine Auftragung der Korngrößenverteilung im RRSB-Netz (siehe Anhang II) liefert eine steilere Steigung als jene der davor verwendeten Verteilung, was zu einem höheren Gleichmäßigkeitsparameter n führt ($n = 2,88$). Der Lageparameter dp' liegt mit 3,5 mm ebenfalls vergleichsweise niedriger. Beide Werte, sowie die Korngrößenanalyse können Tabelle 13 entnommen werden. Die spezifische Oberfläche ergibt sich ebenfalls aus dem Wahrscheinlichkeitsnetz, allerdings wird hierbei zur Berechnung nach Formel 2-14 ein anderer Formfaktor miteinbezogen. Dieser wurde von Voller und Lindthaler mithilfe einer Tabelle auf $f = 1,4$ abgeschätzt, welche sich im Anhang III befindet. [18] Durch den hohen Wert ergibt sich auf folglich eine höhere spezifische Oberfläche. Unter Verwendung des „alten“ Faktors ($f = 1/0,87 = 1,15$) würde das Ergebnis $a_{v,s} = 26,1 \text{ cm}^2/\text{cm}^3$

betragen und somit um rund 18% kleiner sein, was einer viel zu runden Grundstruktur der Körner entsprechen würde.

Tabelle 13: Korngrößenanalyse aus [25] und grafisch ermittelte RRSB-Parameter, sowie geschätzter Formfaktor und die spez. Oberfläche

Korngrößenanalyse		
x [mm]	Q [%]	ΔQ [%]
	100,00	
6,3	82,81	17,19
4,0	58,93	23,89
2,0	23,89	35,04
1,0	2,18	21,71
0,5	0,04	2,14
0,355	0,02	0,02
0,25	0,01	0,01
0,18	0,01	0,00
<0,18		0,01

Parameter aus dem RRSB-Netz		
	Einheit	
n		2,88
d_p	[mm]	3,5
φ ¹⁾		1,4
$a_{v,s}$	[cm ² /cm ³]	31,8

1) Formfaktor wurde geschätzt anhand der Tabelle in Anhang III

Des Weiteren ändert sich in dieser Modellpackung die Porosität. Neuere Untersuchungen ergeben eine viel dichtere Schüttung am Sinterband mit einer Porosität von rund 36 %. Dabei ist aber auch der Feuchteanteil miteinberechnet, welcher rund 4 % ausmacht. Um den Anfangsdruckverlust über die Schüttung zu simulieren wird allerdings trotzdem die Packung mit rund 36 % erstellt.

2.3.2 Erstellung der Modellkörner

Insgesamt werden 17 verschiedene Modelle entwickelt, welche alle auf realen Partikeln aus einer Grünmix-Schüttung basieren. Für die Generierung werden reale Körner aus der Schüttung entnommen und ihre Maße händisch mittels eines Lineals bestimmt. Da es sich um ein einfaches Lineal handelt, wäre ein Abmessen von Partikeln unter 2 mm nicht nur sehr mühsam, sondern auch sehr ungenau. Deswegen sind alle realen Referenzkörner mindesten 2 mm in ihrer größten Ausdehnung lang. Um das Skalieren zu vereinfachen werden alle Modelle mit einer Normgröße von 10 mm erstellt, wobei sich diese Abmaße im Falle nicht sphärischer Partikel auf den größten Durchmesser oder die Länge des Modelles beziehen. Allerdings ist diese Vorgabe nur ein Richtwert. Beim Setzen zufälliger Eckpunkte an eine definierter Grundform kann es passieren, dass zwei dieser Punkte nicht an den am weitesten voneinander gelegenen Orten positioniert werden. Dadurch entsteht ein Gebilde, welches nicht als größten Durchmesser die vorgegebenen 10 mm besitzt. Dies ist insbesondere bei einer Anzahl unter 10 Eckpunkte möglich. Es ist leider nicht möglich, die Abmaße des generierten Korn numerisch mittels der Software zu bestimmen. Stattdessen wird, um zu große Abweichungen von den vorgegebenen 10 mm zu verhindern, das dreidimensionale Gebilde solange gedreht bis optisch die längste Seite vorliegt. Bei der optischen Darstellung ist immer ein Referenzmaß angegeben, welches in Folge zur Abschätzung Korngröße benutzt wird. Des Weiteren werden die Modelle auf einen kleineren Durchmesser skaliert, wodurch sich auch mögliche Abweichungen reduzieren. Abbildung 19 zeigt oben das Abmessen des

realen Referenzkorns und darunter die erzeugte digitale Version. Obwohl durch die Generierung viele Unebenheiten auf der Oberfläche geglättet werden und die Seitenkanten stattdessen vereinfachend Geraden bilden, stellt das Endergebnis eine überraschend gute Annäherung der Realität dar. Von den 17 Modellen sind rund drei einer polyedrischen Kugel nachempfunden und sechs ähneln einer Ellipse. Der Rest sind stärker von der sphärischen Form abweichende Strukturen, wie Tetraeder oder Plättchenformen.

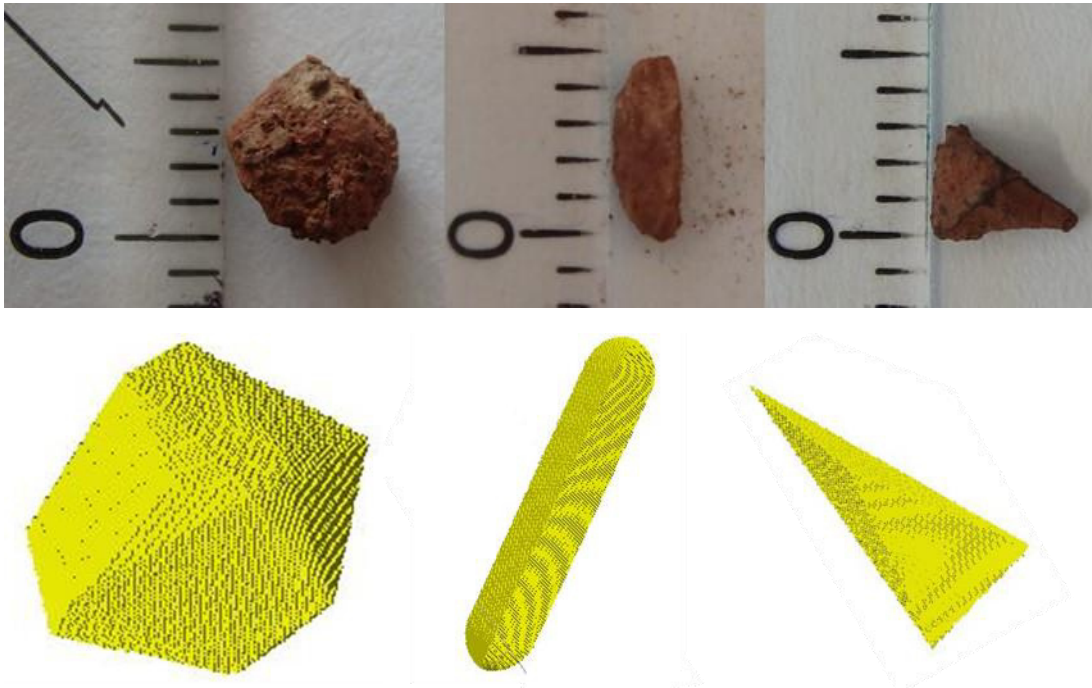


Abbildung 19: Exemplarische Darstellung von Modellpartikeln

2.3.3 Prinzip bei der Packungserstellung

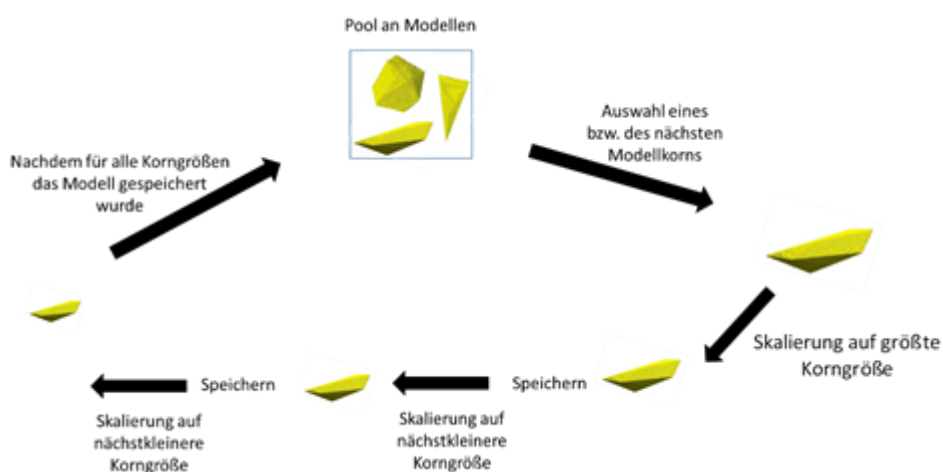


Abbildung 20: Erste Schleife bei der Erstellung der Packung, in der jedes Modellpartikel auf jede Kornklasse skaliert und jeweils abgespeichert wird

Die Basis bei der Generierung der Packung bilden die nichtsphärischen Modellkörner, welche digitale Versionen realer Partikel in einer Schüttung darstellen. Jedes Korn wird nach seiner zufriedenstellenden Generierung einzeln abgespeichert, wobei dessen Platz im Raum oder seine Ausrichtung keine Rolle spielen. Wichtig, neben der Korngröße und möglichst realgetreuen Wiedergabe der Struktur der Referenzpartikel, ist allerdings die Farbe bzw. seine „Material ID“, sowie die Dimensionen des Gebiets, in welchem es erstellt wird. Bei der „Material ID“ handelt es sich laut [13] um einen vierstelligen Code aus Nullen und Einsen (z.B. 0010), mit welcher Materialphasen unterschieden werden können. Jedem Code wird zur Abkürzung einfach einer Nummer zugeschrieben, zum Beispiel handelt es sich bei (0000) um die Material ID 0. Alle Modellpartikel der Modellbibliothek müssen dieselbe Farbe, sowie eine einheitliche Maximalgröße von 10 mm besitzen. Diese Modellpartikel müssen in einem Quader mit Abmaßen (500x500x896 Voxel bzw. 20x20x35 mm) eingebettet vorliegen.

Gesammelt in einem Ordner kann mit der ersten Schleife begonnen werden mit welcher ein Pool aus skalierten Körnern entstehen soll. Dafür wird das erste Modellpartikel geladen und auf die gewünschte Korngröße skaliert. Dabei wird allerdings nicht die untere oder obere Kornklassengrenze als Wert herangezogen, sondern der arithmetische Mittelwert mit einer zufälligen Schwankung von $\pm 20\%$ um diesen. Letzteres wird automatisch generiert, sodass nur der Mittelwert je Kornklasse eingegeben werden muss. Des Weiteren wird die Material ID vor dem Abspeichern geändert, wobei der Grund dafür auf der nächsten Seite erklärt wird. Danach erfolgen ein abermaliges Laden des ursprünglichen Modells und eine Skalierung auf die nächste geforderte Korngröße, sowie das Umfärben vor dem Speichern. In der Reihenfolge wird dabei mit der größten Kornklasse begonnen und danach alle Größen für ein Modell modelliert, bevor schließlich das nächste geladen wird um auch für dieses alle Größen zu erzeugen. Die Schleife endet, sobald für jede Kornklasse und jedes Modellpartikel eine eigene Datei abgespeichert und somit eine Art Bibliothek angelegt wurde.

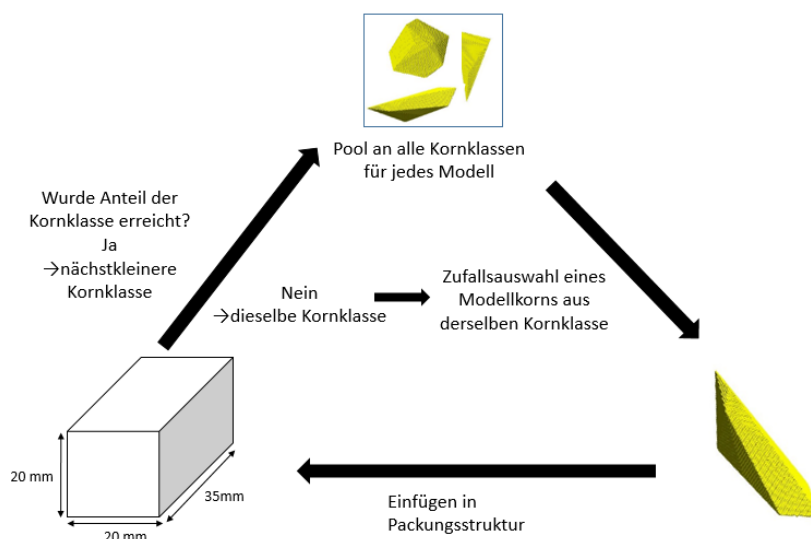


Abbildung 21: Zweite Schleife bei der Erstellung der Packung, in der mit der größten Kornklasse beginnend nacheinander die skalierten Modellpartikel eingefügt werden

Danach beginnt die zweite Schleife, bei der die eigentliche Generierung der Packung erfolgt durch Einfügen der skalierten Partikel in den vordefinierten Bereich. Dieser wurde bereits während der Erstellung der Modellpartikel festgelegt. Da ein Platzieren von Körnern nur in eine bereits geladene Struktur möglich ist, wird zu allererst ein sogenanntes Startpartikel aufgerufen. Dabei handelt es sich um das Modell Nummer eins, welches auf die größte Kornklasse skaliert ist. Beginnend mit der größten Kornklasse erfolgt danach das Hinzufügen der Partikel, wobei diese aus der Modellbibliothek willkürlich ausgewählt werden. Die Platzierung, sowie Ausrichtung sind ebenfalls zufällig. Dabei kann es allerdings zu gegenseitigen Überlappungen kommen, welches wiederum die Korngrößenverteilung verfälschen würde. Die Material ID hilft in diesem Fall nicht nur Überlappungen zu erkennen, sondern diese auch zu entfernen. Platzierte Körner besitzen anfangs die Material ID 03 und solange diese keine anderen Partikel komplett oder auch nur partiell abdecken, wird sie in Material ID 01 geändert. Schneiden sich allerdings beide Farben werden ihre Codes laut [11] bitweise addiert, wodurch die Material ID 02 entsteht. Das „Entfernen“ des unerwünschten Kornes erfolgt einfach durch ändern der ID, das heißt aus der Material ID 02 wird ID 01 und aus Material ID 03 wird ID 0 (Luft). Abbildung 22 zeigt schematisch anhand zweier Kreise das Funktionsprinzip. Durch diese bitweise Summation ist allerdings eine unterschiedliche Färbung der Kornklassen zur Unterscheidung untereinander nicht möglich. Bei acht verschiedenen Korngrößen ergäbe dies viele unterschiedliche Material ID's im Falle einer Überlappung, deren Codierung schwer vorherzusehen sind. Eine Addierung aus ID 02 und ID 04 ergibt beispielsweise ID 06, während aus ID 03 addiert mit ID 05 infolge der Binäraddition die Material ID 07 entsteht. Problematisch sind allerdings jene Additionen, welche als Resultat eine der bereits bestehenden Partikel ID's ergeben. Als Beispiel (entnommen aus [9]) sei hier das Ergebnis aus ID 01 und ID 05 genannt, wodurch wieder ID 05 entstehen würde. Dadurch wäre der Unterschied zwischen einem Partikel und einer Überlappung nicht mehr ersichtlich und könnte auch nicht auf die vorgestellte Weise entfernt werden. Aus diesen Gründen besitzen in GeoDict [9] alle Partikel nach ihrem überlagerungsfreien Einfügen dieselbe Material ID.

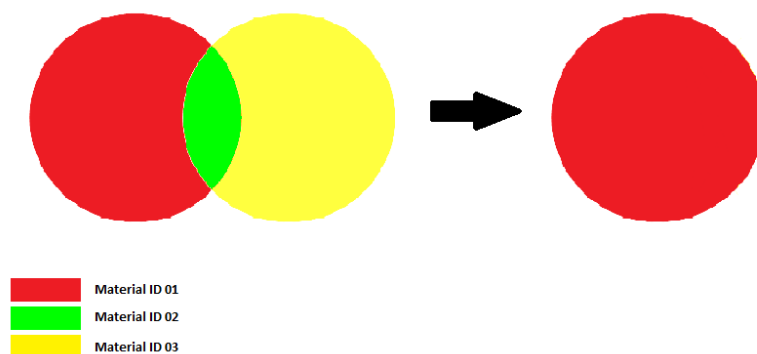


Abbildung 22: Schematische Darstellung der „Entfernung“ einer Überlappung zweier Körner

Auf diese Art und Weise werden die Partikel derselben Kornklasse hinzugefügt bis der dazugehörige, vorgegebene Anteil erreicht wird. Dann beginnt die Schleife wieder mit der nächst kleineren Kornklasse. Um begleitend die Anteile jeder Kornklasse überprüfen zu können, wird zusätzlich der Anteil in einem Textfile abgespeichert, bevor die nächste Größenkategorie eingefügt wird. Des Weiteren wird nach dem Eintrag die bisherige Packung abgespeichert.

Das Einfügen der einzelnen Partikel wäre händisch allerdings viel zu aufwendig. Daher wird der Prozess automatisiert mittels eines Makros. Dabei handelt es sich um eine beliebige Abfolge von Befehlen, mit dem durch einen einfachen Aufruf mehrere Module der Software ausgeführt werden können. Um ein Makro zu codieren muss eine bestimmte Programmiersprache verwendet werden, welche auch von der Software interpretiert werden kann. GeoDict bietet neben einer eigenen Programmiersprache GMC die Möglichkeit ein Makro mittels Python 2.7 zu schreiben. Letzteres ist eine etablierte Sprache, für welche es viele Hilfeforen und Anleitungen gibt. So können die zuvor beschriebenen Schleifen ohne großen Aufwand ausgeführt werden. Um das Makro zu exekutieren müssen in der Software folgende Befehle ausgeführt werden: „Macro“ → „Execute Parameter Macro/Script“. Danach kann das Makrofile mittels „Browse“ ausgewählt und durch einen Klick auf „Run“ gestartet werden. Zu Beginn müssen die Kornklassen, die dazugehörigen Kornanteile, sowie die gewünschte Endporosität angegeben werden. Danach erfolgt die Packungsgenerierung von selbst. Als weiteres Extra kann außerdem der Zielordner benannt werden, in dem die fertige Packung gespeichert wird. Dafür muss vor dem eigentlichen Start des Makros „Edit“ neben „Fixed Parameters“ angeklickt werden. Das Makro ist in Anhang IV gelistet.

Ein großer Nachteil dieses Makros ist seine extrem langsame Geschwindigkeit bei der Packungsgenerierung. So musste der erste Versuch zur Fertigstellung des Modells nach 17 Tagen abgebrochen werden, da der Feststoffanteil gerade einmal 23,47 % (Sollwert: 64 %) erreicht hatte. Der Grund liegt in der willkürlichen Platzierung der Körner in dem Raum. Mit steigender Partikelanzahl wird die Wahrscheinlichkeit immer geringer, dass ein freier Platz gefunden werden kann. Das Programm benötigt somit stetig mehr Versuche und damit mehr Zeit. Eine Auswertung der Porosität über die Zeit ergab einen asymptotischen Verlauf, welcher das Makro sehr ineffizient macht. Auch eine Halbierung des Feststoffanteils bei gleichzeitig doppeltem Volumen führt zu keinem verwertbaren Ergebnis. Obwohl der Trick bei Modell „Polyeder-46“ zum Erfolg führte, musste auch hier die Generierung nach 8 Tagen und einem Feststoffanteil von 19,28 % abgebrochen werden. Nach der allseitigen Komprimierung würde zwar der Feststoffanteil bei 38,56 % liegen, zur Fertigstellung würde das Programm allerdings noch Wochen brauchen. Als Extremfall wurde der Feststoffanteil auf ein Drittel minimiert und das Volumen verdreifacht, womit die zu erreichende Sollporosität bei 78,67 % wäre. Zusätzlich wurde die Anzahl der Kornklassen auf drei reduziert. Aber auch diese Maßnahmen führten nach zwei Wochen zu keinem Ergebnis. Die Generierung wurde bei einer Porosität von 82,75 % abgebrochen und versuchsweise komprimiert auf die gewünschten Dimensionen. Die einzelnen Körner zeigen allerdings heftige Verzerrungen an der Oberfläche und verschmelzen stark ineinander. Obwohl das Makro grundsätzlich funktioniert, ist sein Algorithmus zu

ineffizient. Infolge dessen wurde ein alternatives Programm entwickelt, welches nun nachfolgend vorgestellt wird.

Effizientere Programmierung

Das neue Programm „PackageCreator“ baut auf einer C++-basierten Umgebung auf und ist fast unabhängig in seiner Packungsgenerierung von dem Programm GeoDict. Es benötigt vor dem eigentlichen Start im Wesentlichen nur die Modellpartikel, welche zuvor in GeoDict angefertigt und für „PackageCreator“ in Form von ASCII-Dateien zur Verfügung gestellt werden. In diesen sind die Voxel, die das Korn aufbauen als Nullen und Einsen hinterlegt, wobei 1 der Material ID eines Feststoffvoxels entspricht und 0 ein Voxel aus der Partikelumgebung darstellt. Im geöffneten Zustand sind in der Datei in der ersten Zeile die Dimensionen in Voxel für die X-, Y- und Z-Richtung zu sehen. Danach beginnt eine Matrix in der die YZ-Ebene Schicht für Schicht entlang der X-Achse aufgezeichnet ist. Das Programm ist nur in der Lage genau diesen Dateiaufbau zu lesen. Des Weiteren wird eine Parameterdatei in Form eines .txt-Formates benötigt, welche die wichtigsten Eckdaten beinhaltet. Auch hier muss der Aufbau einer bestimmten Abfolge entsprechen, wobei dieser dem Dateilisting in Anhang V entnommen werden kann. Für die Eingabe der Kornklassengrößen wird auch hier wieder der Klassenmittelwert herangezogen mit einer Schwankung von $\pm 20\%$, welche von dem Benutzer selbst eingestellt werden. Relevant ist darin die Eingabe der Ausgangsgröße als erste Kornklasse mit einem Anteil von Null. Der Wert wird später für die Skalierung verwendet. Wichtig ist ebenfalls die Benennung dieser Anfangsdateien, so dürfen für eine exakte Erkennung der Modelle diese ausschließlich mit „Particle_[Modellpartikelnummer].leS“ beginnen. Die eckige Klammer stellt hier einen Platzhalter dar, in welchen die Nummer des Modells eingetragen werden soll. Auch die Datei, welche die Parameter beinhaltet, darf ausschließlich mit „Parameter.txt“ benannt werden.

Bei der Erstellung der Modellkörner sind einige Dinge zu beachten. Grundsätzlich müssen bei „PackageCreator“ im Gegensatz zu dem vorherigen Python-basierten Makro die Dimensionen der Modelle nicht mehr mit jenen der endgültigen Packung übereinstimmen. In diesem Fall ist es sogar sinnvoller den Leerraum um das Modellkorn herum zu minimieren, da die Dateigröße dabei stark reduziert wird und somit leichter bzw. schneller zu händeln ist. Die Boxgröße ist somit beliebig je nach Modell wählbar. Auch die Material ID der Modelle ist hierbei nicht relevant und nimmt keinen Einfluss auf die Packungsgenerierung. Im Gegensatz zum Python-Makro darf das Partikel aber in Folge einer Reduktion der Seitenlänge nicht mehr über den Seitenrand hinausragen und auf der gegenüberliegenden Seite fortgeführt werden, sprich periodisch sein. Der Algorithmus kann ansonsten das Korn in seinem Ganzen nicht erfassen. Auch die Ausgangsgröße des Modells ist sowohl nach unten, als auch nach oben begrenzt. Bei dem Einfügen, Skalieren und Rotieren kann es grundsätzlich zu Abweichungen im gewünschten Durchmesser kommen, welche größer sind, je kleiner der Skalierungsfaktor ist. Dieser Effekt wird verstärkt, wenn die Ausgangsgröße zu klein ist. Jedoch bei einer zu starken Erhöhung des Durchmessers kommt es zu einem Absturz des Programms, weswegen für die nachfolgende Packungsgenerierung eine Ausgangsgröße von 15 mm gewählt wird. Dadurch

können die Abweichungen bis zu einer Kornklasse von 1 mm mit maximal $\pm 5\%$ in Grenzen gehalten werden.

Durch einen Doppelklick auf die „PackageCreator.exe“-Datei wird die Packungsgenerierung gestartet, wobei zuerst die Befehlszeile geöffnet wird. In dieser können zusätzliche Optionen ausgewählt werden, auf die aber erst später eingegangen wird. Mit der Eingabe von „0“ wird das Programm gestartet, wobei der erste Schritt die Initialisierung des Raumes ist, in welchen später die Körner eingeschichtet werden. Anschließend werden die Kornklassenanteile in der Befehlszeile ausgegeben, wobei es sich dabei nicht um die in die Parameterdatei eingegebenen Kornklassenanalyse handelt. Der angezeigte Prozentsatz bezieht sich zum einen auf den gesamten Raum und stellt zum anderen den summierten Anteil dar. Dann beginnt die Initialisierung der Modellkörner, indem jedes Partikel nacheinander einzeln aufgerufen und geöffnet wird. Da in der ersten Zeile der Dateien jeweils die Anzahl der Voxel für jede Dimension enthalten sind (siehe Anhang V) werden die Werte je einer neuen Variable zugewiesen. Danach liest das Programm die restlichen Zeilen Schritt für Schritt durch und speichert dabei jedes Feststoffvoxel in Form eines Vektors ab. Aus der Gesamtheit dieser Vektoren wird anschließend der Schwerpunkt des Partikels berechnet, indem alle Vektoren aufsummiert und diese durch die Summe der Vektoren dividiert werden. Gleichzeitig werden die sechs äußersten Punkte des Modells (Minimal- und Maximalwert in jeder Dimension) in einer eigenen Liste gespeichert. Das Ergebnis aus der Schwerpunktberechnung (wieder ein Vektor) bildet ab sofort den neuen Ursprung und ersetzt damit den Koordinatenursprung. Nachdem diese Schrittfolge für alle Partikel durchlaufen ist, beginnt die eigentliche Schichtung der Körner in eine Packung.

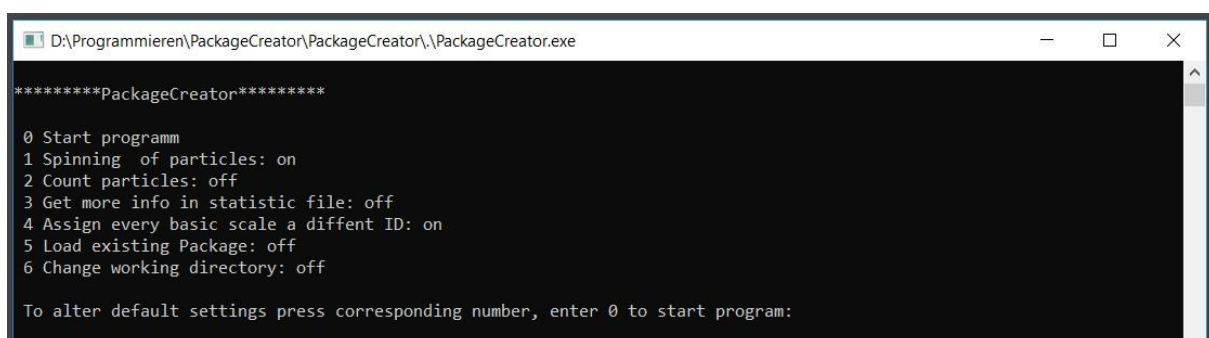
Die Initialisierung zum Füllen des leeren Raumes beginnt damit, dass der Algorithmus die größte Kornklasse aufruft und den Kornklassenanteil in der Packung überprüft. Solange der gewünschte Anteil nicht erreicht wurde, schichtet das Programm weiter Körner derselben Kornklasse ein. Ist die Grenze allerdings erreicht, wird die nächst kleinere Kornklasse aufgerufen. Dann wird ein zufälliger Vektor, welcher innerhalb des noch leeren Raumes liegt, generiert und die gewählte Stelle auf „0“ oder nicht „0“ überprüft, das heißt ob der Platz frei ist oder nicht. Ist der Vektor schon belegt, erfolgt die Generierung eines neuen zufälligen Vektors. Standardgemäß handelt es sich bei dem ersten, einzufügenden Modell um die Nummer 1, was auch beim Python-Makro der Fall ist. Im späteren Verlauf erfolgt die Auswahl dann ebenfalls zufällig. Für das Modell wird ein Rotationsparameter anhand Formel 2-18 [3], sowie ein Skalierungsfaktor berechnet, welcher innerhalb des gewünschten Schwankungsbereiches um den Mittelwert der jeweiligen Kornklasse liegt.

$$R_{\hat{n}}(\alpha)\vec{x} = (1 - \cos \alpha)\hat{n}(\hat{n} \cdot \vec{x}) + \cos \alpha \vec{x} + \sin \alpha (\hat{n} \times \vec{x}) \quad \text{Formel 2-18}$$

Im nächsten Schritt werden die abgespeicherten Eckpunkte des Modells nacheinander aufgerufen und jeder dieser Eckpunkte mit den vorhin errechneten Parametern rotiert, skaliert und dessen Schwerpunkt mit dem zufällig erstellten Vektor addiert. Somit erhält das Korn seine neue Position im Raum. Da es sich dabei um einen periodischen Raum handeln soll, erfolgt noch eine Korrektur für den Fall, dass das Partikel über den Rand hinausragt. Zum

Schluss wird die neue Position jedes Eckpunktes auf eine Nullstelle überprüft. Nach erfolgreicher Inspektion der Eckpunkt-Positionen wird der Vorgang des Rotierens, Skalierens und Vektoraddierens mit denselben Parametern mit den restlichen Feststoffvektoren des Korns durchgeführt. Ist auch in diesem Fall der gesamte Platz frei, so werden erst dann im Raum die Nullstellen durch die gewünschte Material ID ersetzt. Bis zu diesem Zeitpunkt rechnet und überprüft der Algorithmus nur theoretisch die mögliche Position in der Packung. Außerdem wird erst nach einem erfolgreichen Einfügen das nächste, willkürlich gewählte Modell ausgewählt. Für den Fall, dass die Position schon belegt ist, erfolgt immer wieder eine Neuberechnung des zufälligen Vektors, Rotationsparameters und Skalierungsfaktors für dasselbe Modell, bis dieses einen Platz gefunden hat. Dann wird noch die gesamte Partikelanzahl im Raum gezählt und in der Befehlszeile neue Informationen angezeigt. Diese beinhalten, unter anderem, den derzeitigen, gesamten Feststoffgehalt, Partikelanzahl, usw. Für nähere Informationen dazu wurde ein beispielhafter Ausschnitt während einer Packungsgenerierung in Anhang V angefügt. Sobald die Ausgabe erfolgt ist, beginnt der Algorithmus wieder von vorne mit der Überprüfung des gesamten Feststoffgehaltes und ob dieser bereits den gewünschte Summen-Kornklassenanteil überschritten hat. Für das Programm sind also die angegebenen Zahlen für den gesamten Feststoffgehalt und der Kornklassenanteile in der Parameterdatei Werte, welche mindestens erreicht werden sollen. Sie können demnach eventuell überschritten werden. Sobald der geforderte, gesamte Feststoffgehalt erreicht wurde, schließt das Programm mit der Ausgabe einer Statistik-Datei und der fertigen Packung in der ASCII-Codierung ab. In ersteren sind die wichtigsten Eckdaten während des Generierungsprozesses aufgelistet. Für Details ist in Anhang VI ein beispielhafter Ausschnitt der Statistik-Datei des fertigen Modells eingefügt. Da es sich hierbei um einen komplexen Ablauf handelt, wird ein übersichtliches Fließdiagramm in Abbildung 24 dargestellt. Eine Arbeitsanleitung ist im Anhang VII und der Programmcode im Anhang VIII zu finden.

Wie kurz zuvor erwähnt, erscheint zu Beginn in der Befehlszeile eine Reihe an zusätzlichen Optionen, welche hinzugefügt oder abgeschaltet werden können. In Abbildung 23 ist ein Screenshot des Startfensters zu sehen und die unterschiedlichen Möglichkeiten, welche danach näher erklärt werden.



```
D:\Programmieren\PackageCreator\PackageCreator\.\PackageCreator.exe
*****PackageCreator*****
0 Start programm
1 Spinning of particles: on
2 Count particles: off
3 Get more info in statistic file: off
4 Assign every basic scale a different ID: on
5 Load existing Package: off
6 Change working directory: off

To alter default settings press corresponding number, enter 0 to start program:
```

Abbildung 23: Befehlszeile nach dem Starten des Programms

0. Start programm = „PackageCreator“ startet.
1. Spinning of particles = Partikel werden beim Einfügen in eine zufällige Orientierung gedreht.
2. Count particles = In der Korngrößenverteilung in der Parameter-Datei wird bei „on“ nicht der Kornanteil als Volumenprozent gelesen, sondern als absolute Partikelanzahl pro Klasse.
3. Get more info in statistic file = Zusätzliche Informationen, z.B. die Positionen der Schwerpunkte aller eingefügten Partikel oder die Rotationsparameter werden in die Statistik-Datei zum Schluss hinzugefügt.
4. Assign every basic scale a different ID = Jeder Kornklasse wird eine andere Material ID zugewiesen
5. Load existing Package = Anstatt einen leeren Raum zu generieren, kann hiermit eine bereits bestehende Packung geladen werden
6. Change working directory = Der „PackageCreator“ verwendet jene Modelle, welche sich in denselben Ordner befinden. Sind diese jedoch wo anders gespeichert, kann mittels dieser Zusatzoption das Verzeichnis mit den darin enthaltenen Modellen ausgewählt werden.

Obwohl es sich hierbei um einen sehr effizienten Einschichtungs-Algorithmus handelt, ist auch dieses Programm hinsichtlich des maximalen Feststoffgehaltes begrenzt. In Folge der Generierung einer Packung mit nur 36 % Luftanteil wurde festgestellt, dass dies nur sehr schwer möglich ist. Es wird daher empfohlen, ein Modell mit einem Feststoffgehalt von maximal 50 % herzustellen. Um trotzdem die gewünschten Porosität von 36 % zu erreichen, werden wie bereits bei dem Python-Makro die Körner gemäß Korngrößenverteilung nach Tabelle 13 in einen größeren Bereich eingeschichtet und nach erfolgreichem Einfügen dieser komprimiert. Wie schon zuvor erwähnt, kommt es durch die Rotation und Skalierung der Partikel zu Voxelfehler, was zu Abweichungen des gewünschten Durchmessers führt. An der Oberfläche der einzelnen Körner entstehen leichte Verzerrungen. Als das Volumen für das Einfügen von 400 x 400 x 700 Voxel auf 500 x 500 x 896 Voxel verdoppelt wurde, zeigte sich nach dem Komprimieren eine starke Intensivierung dieser Verzerrungen. Um diesen Effekt zu vermindern, besaß das fertige Modell vor seiner Komprimierung Dimensionen von 450 x 450 x 802 Voxel. Damit ist das Volumen nur 1,45-mal so groß, wie der gewünschte Sollwert und der Feststoffgehalt konnte um diesen Faktor auf 44,14 % für die Generierung reduziert werden. Die Seitenlängen mussten damit nur um etwa 12,72 % in Z-Richtung und die restlichen Längen um nur 11,11 % komprimiert werden. Das Ergebnis zeigt nur leichte Verzerrungen und ist in nachfolgend detaillierter beschrieben.

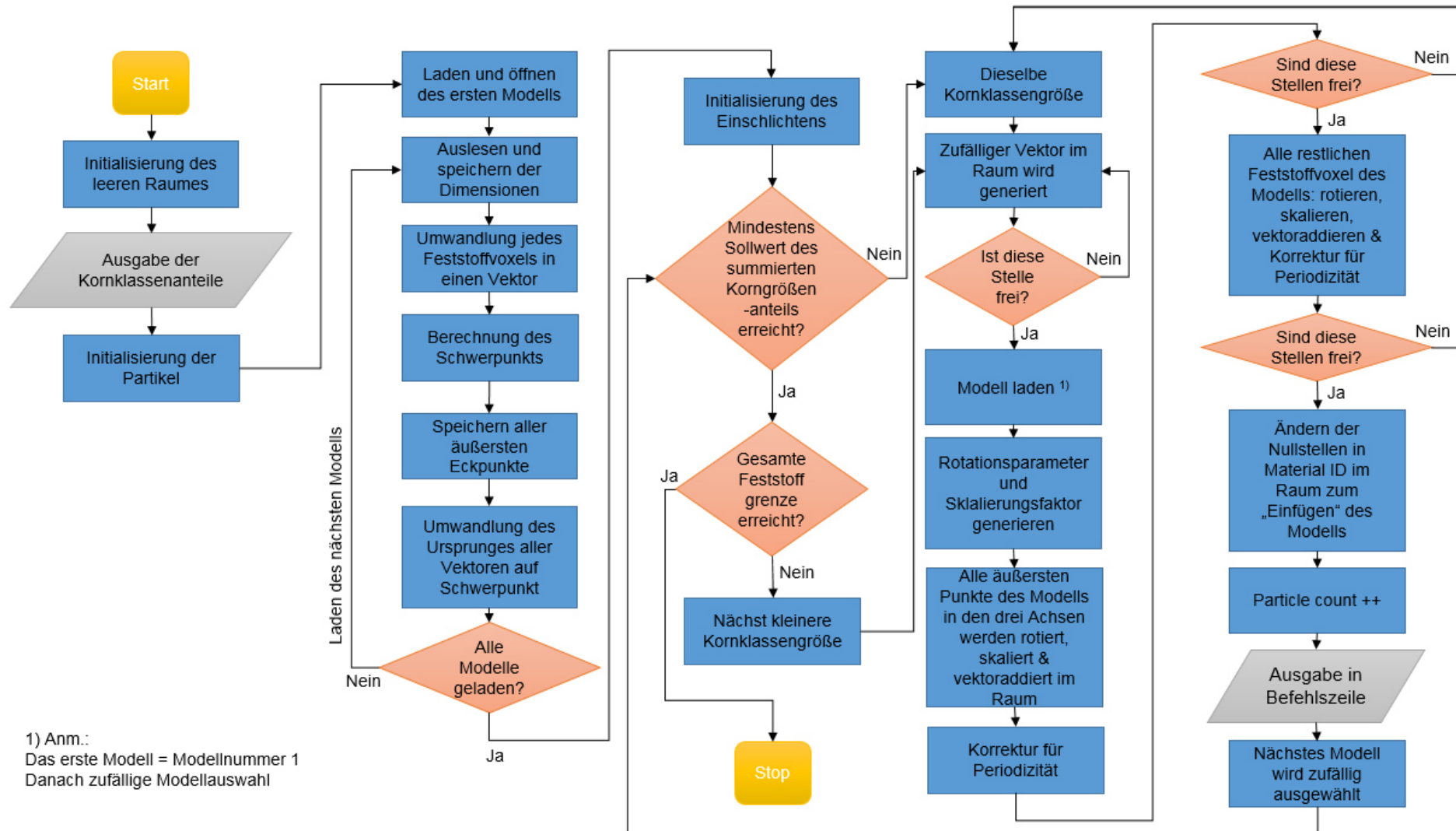


Abbildung 24: Ablauf des „PackageCreator“ in Form eines Fließdiagrammes

2.3.4 Präsentation der fertigen Packung mit Kennparametern

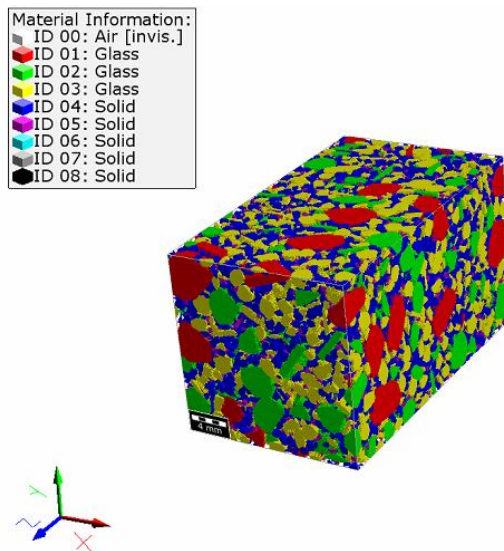


Abbildung 25: 3D-Darstellung der fertigen Packung aus den selbst erstellten Modellformen mit einer Korngrößenverteilung aus [25] in einer Box mit den Abmaßen von 20x20x35 mm und einer Voxelgröße 50 µm

Dieses Modell besitzt Abmaße mit 400 x 400 x 700 Voxel und eine Voxelgröße von 50 µm. Dank des neuen effizienteren Algorithmus im „PackageCreator“ dauerte die Generierung dieser Packung etwa fünfeinhalb Stunden. Da es sich bei der fertigen Datei um eine .leS-Datei bzw. ASCII-Datei handelt, muss bei dem Import des Modells die Voxelgröße selbst eingegeben werden. Da das vorherige mehreckige Modell aus Kapitel 2.2 bereits ein erstes repräsentatives Modell für die vorgegebene Korngrößenverteilung ist, werden dessen Dimensionen und Auflösung auch hier übernommen. Der Sollwert für die Porosität von 36 % wurde auch annähernd erreicht. Die in Abbildung 25 dargestellte, dreidimensionelle Packung besitzt einen Luftanteil von 35,99 % und jede Kornklasse ist mit einer anderen Material ID versehen. Die ersten drei sind laut Legende mit dem Material Glas hinterlegt (Voreingestellte Benennung von GeoDict, bei Bedarf änderbar). Auch für die darauffolgenden Strömungssimulationen spielt die Art des Materials keine Rolle, solange dieses als Feststoff eingeordnet ist. Die darin befindlichen Partikel sind nach der Korngrößenverteilung in Tabelle 13 modelliert, wobei sich auch hier während der Generierung Abweichungen ergaben. Die tatsächlich erreichten Werte sind in der nachfolgenden Tabelle 14 aufgelistet. Die Packung besteht aus einer Bandbreite aus Formen (insgesamt 17 verschiedene), welche alle selbst modelliert und während dem Einschichten zufällig bestimmt und an eine beliebige Stelle im Raum gesetzt wurden. Jede Kornklasse besteht somit aus einer zufällig generierten Mischung der Formen. Da das Programm auch die Modellnummer jedes eingefügten Partikels mitschreibt, kann eine Statistik darüber erhoben werden. Die Ergebnisse daraus sind ebenfalls in der Tabelle enthalten.

Tabelle 14: Parameter und Eigenschaften der erstellten Packung aus 17 verschiedenen Partikelform-Modellen

Packungsparameter			
NXxNYxNZ [Anzahl]	400x400x700		
Voxelgröße	50	[µm]	
Quader [mm]	Fläche	20x20	
	Höhe	35	
Material	Glas		
	Feststoff		
	Pore	Luft	
Porosität	Ist	Soll	
	35,99	36	[%]
Kornklassen [mm]	Mittelwert [mm]	Ist [%]	Soll [%]
6,3-10	8,15	17,75	17,19
4-6	5,15	23,36	23,89
2-4	3	35,02	35,04
1-2	1,5	21,70	21,71
0,5-1	0,75	2,14	2,14
0,355-0,5	0,4275	0,02	0,02
0,25-0,355	0,3025	0,02	0,01
0-0,25	0,125	0,02	0,01

2.4 CT-basierte Sinterstruktur

Neben der Erstellung von Grünmix-Packungen durch stochastisches Anordnen von Modellpartikeln, wie es auf den vorherigen Seiten beschrieben wurde, sollen auch die Möglichkeiten und Grenzen der Packungsgenerierung ausgehend von dreidimensionalen Sinterstrukturen ausgetestet werden. Als Basis wird der CT-Scan eines fertigen Sinterstücks betrachtet, welcher allerdings vorher bearbeitet wurde. Die Dateigröße des ursprünglichen Gebildes beträgt über 20 Gigabyte und kann somit nicht direkt hochgeladen werden. Des Weiteren ist die äußere Form so unregelmäßig, dass eine anschließende Durchströmung zu keinem verwertbaren Ergebnis geführt hätte. Der simulierte Luftstrom wäre außen am Feststoff vorbeigeflossen, anstatt diesen zu durchströmen. Deswegen wurde das Gefüge von allen Seiten planparallel geschnitten bis außen keine durchgängigen Luftkanäle mehr vorhanden sind.

Das Ergebnis ist schließlich ein annähernd kubisches Gebilde mit Abmaßen von 3,58 x 3,59 x 3,44 mm und einer Voxelgröße von 17,1 µm. Für eine einfache Kurzbeschreibung wird dieser Ausschnitt des Sinterstücks im nachfolgenden als Würfel bezeichnet. Abbildung 26 zeigt ein dreidimensionales Bild der porösen Struktur nach dem erfolgreichen Import in GeoDict. Der Grad der Auflösung entstammt dabei aus dem CT-Scan und muss beim Hochladen angeführt werden. Trotz teilweiser großer Poren nimmt der Hohlraum nur rund 23,6 % des gesamten

Volumens ein, womit die Struktur bereits sehr kompakt erscheint. Dem Material wird hierbei im Gegensatz zu den vorhergehenden Modellpackungen kein bestimmtes Material zugeschrieben, sondern nur die Material ID 01.

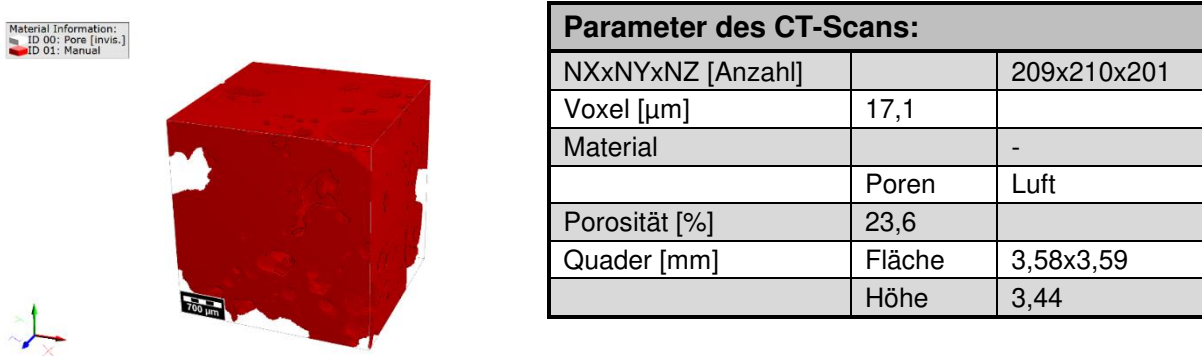


Abbildung 26: 3D-Modell eines würfelförmigen Ausschnittes aus dem CT-Scan eines Fertigsinterstücks mit Abmaßen von 3,58x3,59x3,44 mm und einer Voxelgröße von 17,1 µm

2.4.1 Modifikation des Sinters

Das Material bzw. seine Material ID kann für den importierten Würfel einfach verändert werden. Des Weiteren lassen sich die Dimensionen nach Belieben verkleinern oder vergrößern, wobei sich für letzteres eine Spiegelung oder Wiederholung der Struktur in jede beliebige Richtung anbietet. Auch die Struktur selbst lässt sich modifizieren. Die Software bietet die Funktion „Add Binder“ an, dessen Algorithmus laut [15] ein Bindematerial in Form eines konkaven Meniskus dort hinzufügt, wo die Oberflächen der Struktur nah beieinander sind. Die Menge an Bindematerial hängt dabei von dem eingegeben Feststoffanteilstuwachs ab. Da allerdings nur ein Hinzufügen von festem Material möglich ist, muss die Struktur zuvor invertiert werden.

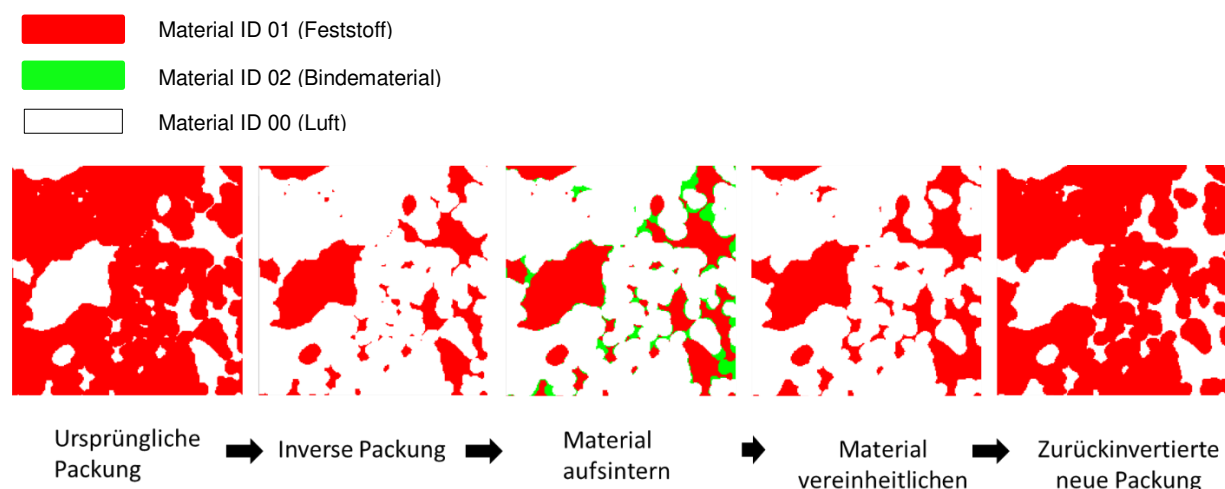


Abbildung 27: Schritte zur Modifikation der Sinterstruktur um virtuell deren Porosität zu erhöhen

Die Schrittabfolge zur Modifizierung der Sinterstruktur wird zum besseren Verständnis in Abbildung 27 dargestellt. Vor der Anwendung der Funktion müssen der Feststoff und der Porenraum invertiert werden (Bild 2), das heißt ihre Material ID's werden vertauscht. Die Poren stellen somit den neuen Feststoff dar, welcher durch das Bindematerial teilweise verbunden wird. Dieses besitzt zur Überprüfung des Vorgangs am Anfang eine andere Farbe (Bild 3). Nach dem erfolgreichen Verändern der Struktur wird diese bezüglich ihrer Material ID vereinheitlicht (Bild 4) und zurückinvertiert (Bild 5).

Dieser Vorgang wird insgesamt viermal an dem im vorherigen Kapitel 3.3.1. vorgestellten Würfel durchgeführt. Dabei werden jeweils rund 5 % an Feststoff „abgetragen“ und somit schrittweise der Volumenanteil von einer Ausgangsporosität von 23,6 % erhöht bis schließlich 45,3 % erreicht sind. Dieser Wert entspricht etwa jener Porosität der im Kapitel 2 vorgestellten Modellpackungen. In Abbildung 28 sind die vier Ergebnisse aus der Modifikation inklusive des ursprünglichen Würfels (ganz links) in dreidimensionaler Ansicht nach steigender Porosität gereiht. Die Differenz der Porositäten entspricht dabei ungefähr jenem Anteil, welcher während des Modifikationsprozesses als Bindematerial addiert wurde (~5 %). Auch die spezifische Oberfläche wird in Folge des Prozesses verändert.

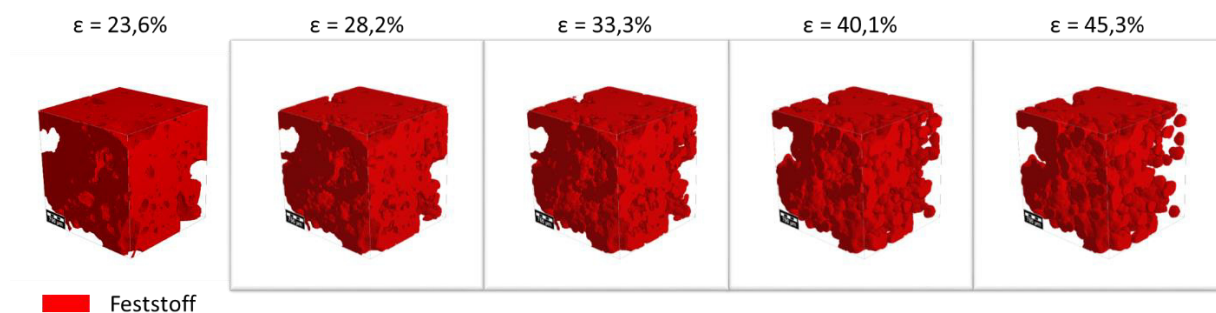
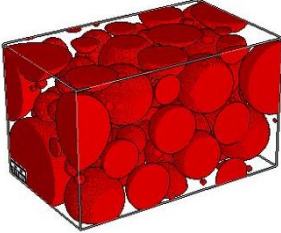
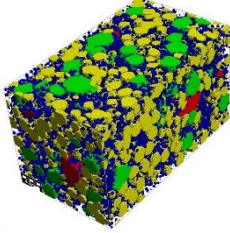
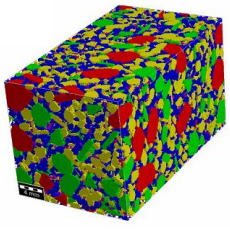


Abbildung 28: 3D-Modell des originalen Sinterwürfels (ganz links) und Modifikationen dazu, aufgereiht nach steigender Porosität um ~5%

2.5 Übersicht der Modelle

In diesem Unterkapitel werden nochmals alle bisher beschriebenen Modelle aus den ersten drei Unterkapiteln zusammengefasst. In der Tabelle 15 sind neben einer dreidimensionalen Darstellung der Packungen, deren wichtigste Eckdaten und Kurzbezeichnungen aufgelistet. Die Benennungen beziehen sich dabei auf die charakteristischen Partikelformen, wobei jene Modelle aus Kapitel 2.2.3 und 2.3.4 zusätzlich den Wert der jeweiligen Porosität beinhalten. Die Kurzbezeichnungen werden auch in der nachfolgenden Arbeit als Abkürzung verwendet.

Tabelle 15: Übersicht aller virtuell erstellten Modellpackungen mit den wichtigsten Kennparametern

Übersicht aller Modelle			
Kurzbezeichnung	„Kugel“	„Polyeder-46“	„Polyeder-36“
			
Porosität [%]	45,94	46,13	35,99
Dimensionen [mm]	14,92 x 14,92 x 24,87	20 x 20 x 35	20 x 20 x 35
Voxelgröße [µm]	49,7	50	50
Kornklassen	16	16	8
Korngrößenbereich [mm]	0,1-12	0,1-12	0,1-9,7 ¹⁾

1) Diese Werte ergeben sich aus dem kleinsten und größten angewandten Skalierungsfaktor

3 Numerische Ermittlung von Packungseigenschaften

Nachdem die im vorherigen Kapitel vorgestellten Modellpackungen erfolgreich generiert wurden, sind an diesen numerische Analysen und Strömungssimulationen durchzuführen. Die Software GeoDict stellt auch hier die dafür notwendigen Modelle und Algorithmen bereit. Es werden unter anderem neue Methoden zur Ermittlung der spezifischen Oberfläche aufgezeigt, deren Anwendung anhand der CT-basierten Sinterstruktur erklärt wird. Des Weiteren werden in diesem Kapitel die mathematischen Hintergründe des angewandten Strömungsmodells, seine Einflussparameter und Bedingungen zum Lösen der Gleichung erläutert. Auch eine detaillierte Beschreibung der empirischen Durchströmungsgleichungen, welche später als Vergleich zu den numerisch berechneten Druckverlustkurven dienen, findet sich im Kapitel 3.3.

3.1 Geometrische Strukturparameter

Die wichtigsten Parameter in den mathematischen Druckverlustgleichungen sind die Porosität und die spezifische Oberfläche bzw. der Sauterdurchmesser. Die angegebenen Ist-Werte für den Luftanteil der unterschiedlichen Modellpackungen ergeben sich aus der Anzahl der Voxel für die Material ID 00 bezogen auf die Gesamtanzahl aller Voxel. Dieser Material ID wird das Material Luft zugeordnet. Die Porosität ist somit nicht nur abhängig von der Menge an eingefügten Partikeln, sondern auch von der Auflösung des Raumes. Wie später noch detaillierter in Kapitel 3.3 aufgezeigt wird, haben bereits kleine Schwankungen in den Werten der spezifischen Oberfläche bzw. des Sauterdurchmessers große Auswirkungen auf die berechnete Permeabilität. Aus diesem Grund ist eine Analyse dieser Einflussgrößen mittels unterschiedlicher Methoden von Vorteil, um mögliche Fehler bei dem Prozess der Ermittlung zu vermeiden. Neben der grafischen Bestimmung mittels RRSB-Netz und der Berechnungen aus den Feinheitparametern einer RRSB-Verteilung (siehe Formel 2-7), bietet die Software GeoDict die Möglichkeit einer numerischen Herangehensweise zur Ermittlung der spezifischen Oberfläche einer Packungsstruktur. Diese neuen Analysemethoden werden nachfolgend genauer erläutert.

3.1.1 Numerische Ermittlungsmöglichkeiten

Die polydisperse Schüttung mit seinen unregelmäßig, eckig geformten Partikeln macht eine direkte Berechnung der spezifischen Oberfläche nahezu unmöglich. Die numerische Methode bietet eine approximative Herangehensweise zur Lösungsfindung und hilft womöglich einen exakteren Wert zu ermitteln, als es basierend auf Abschätzmethode mit vereinfacht kugelförmigen Partikeln möglich wäre. Insgesamt stehen drei verschiedene Varianten zur Verfügung, welche auf unterschiedliche Algorithmen basieren: „Sum Voxelfaces“, „Surface Fit“ und „Voxel dilate“.

Ersteres beschreibt nach [14] einen Algorithmus, welcher alle Voxelflächen an der Stukturoberfläche zusammenzählt und mit Hilfe der Voxelgröße die Oberfläche berechnet. Dieser Wert wird schließlich zur Ermittlung der spezifischen Oberfläche herangezogen. Die

Methodik hinter „Sum Voxelfaces“ basiert im Grunde auf ein simples Zählverfahren, wodurch das Ergebnis stark von der Auflösung abhängt. Je kleiner die Voxelgröße, desto besser wird der wahre Wert approximiert. Der „Surface Fit“ stellt im Gegensatz dazu einen deutlich komplexen Ansatz dar. Mithilfe komplexer Berechnungen werden Diagonale durch den Oberflächenvoxel gezogen und somit die Struktur approximiert; [19]. Der letzte Ansatz zur Bestimmung der spezifischen Oberfläche basiert auf einem Algorithmus, welcher die Feststoffoberfläche mit einer Schicht bedeckt, welche nur 1 Voxel dick ist. Anschließend wird die Anzahl dieses Deckmaterials gezählt und mit der Seitenfläche s eines Voxels multipliziert. Die Auflösung und somit auch die Länge eines Voxels beträgt rund $17,1 \mu\text{m}$. Allerdings muss hierbei ein Korrekturfaktor k miteinberechnet werden, da bei dieser Art und Weise pro Voxel nur eine Fläche berücksichtigt wird. Bei der vorherrschenden, komplexen Struktur können aber mehrere Seitenflächen an der Oberfläche liegen. Im Falle einer konkaven Form würden durch die Voxelschicht zu wenig Flächen gezählt. Abbildung 29 stellt eine umhüllende 1-Voxelschicht (blau) um eine zweidimensionale Struktur (rot) dar, nachdem die Funktion „Voxel dilate“ angewendet wurde. Daran ist zu erkennen wie durch die verwundene Oberfläche eigentlich 14 rote Flächen außen liegen würden, aber nur 10 blaue Voxelflächen gezählt werden. Für die Bestimmung des Faktors wird das Ergebnis aus „Voxel dilate“ jeweils mit den Werten aus „Sum Voxelfaces“ und „Surface fit“ verglichen. Daraus ergeben sich zwei Korrekturfaktoren, wodurch je eine der beiden Methoden approximiert werden kann. Diese Ermittlungsmethode umfasst somit die nachfolgenden Formeln, wobei die spezifische Oberfläche selbst dann einfach mittels Formel 2-8 berechnet werden kann.

$$A_s = k * \text{Anzahl der Voxelschicht} * s^2$$

Formel 3-1

$$V_s = (1 - \varepsilon) * V_{ges}$$

Formel 3-2

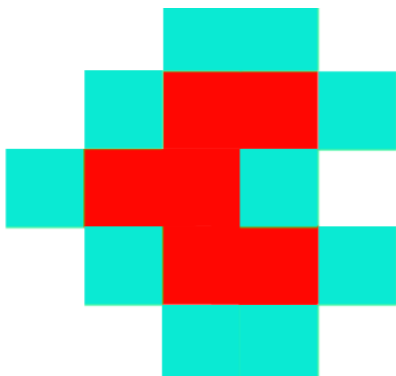


Abbildung 29: Funktion „Voxel dilate“ angewendet an einer exemplarischen, roten Struktur, wodurch diese mit einer blauen 1-Voxelschicht bedeckt wird

Für jede in Abbildung 28 dargestellte Struktur wird die spezifische Oberfläche durch die drei vorgestellten Varianten ermittelt. In Tabelle 16 sind alle Ergebnisse aus der Analyse aufgelistet. Wie erwartet ergibt sich aus dem Surface fit ein geringerer Wert im Vergleich zu Sum of Voxelfaces, wobei die Abweichung über alle Strukturvarianten fast konstant 34 %

beträgt. Beide Methoden liefern somit einen ähnlichen Werteverlauf. Nach den ersten zwei Modifikationen wächst die spezifische Oberfläche bis zu einer Porosität von 33,3 %. Danach verringert sich das Ergebnis wieder und erreicht bei einer Porosität von 45,3 % sogar fast wieder den Anfangswert. Am Anfang der Strukturmanipulation werden gemäß des Algorithmus die fein verteilten Poren verbunden und dadurch die Oberfläche im Vergleich zum Volumen stärker vergrößert. Dadurch steigt der Wert der spezifischen Oberfläche. Mit fortschreitender Feststoffabnahme werden schließlich fast alle inneren Löcher aufgeschlossen und es kommt zu keiner weiteren signifikanten Zunahme der Oberfläche, während der Volumenanteil weiterhin sinkt. Im Gegensatz zu diesen Methoden ergeben sich aus der dritten Variante *Voxel dilate* stetig steigende Werte, wobei die Differenzen mit sinkendem Feststoffanteil abnehmen. Obwohl bei diesen Ergebnissen kein Korrekturfaktor miteinberechnet ist und somit von der erzeugten Deckschicht an der Strukturoberfläche nur eine Fläche pro Voxel gezählt wird, übertrifft die spezifische Oberfläche bereits ab der ersten Modifikation jene Ergebnisse aus *Surface fit*. Durch deren Abnahme der Werte nimmt das Gefälle bei höheren Porositäten stark zu. In einem genaueren Vergleich zwischen *Sum of Voxelfaces* und *Voxel dilate* zeigt sich, dass die Voxelanzahl in beiden Fällen zuerst steigt und schließlich fällt. Der unterschiedliche Werteverlauf mit höherer Porosität ergibt sich schließlich aus der Differenz, denn diese ist bei *Voxel dilate* nicht sehr groß. Der sinkende Feststoffanteil und die daraus resultierende Volumenabnahme sind stärker, als die Abnahme der Oberfläche. Dadurch steigt die spezifische Oberfläche kontinuierlich bei *Voxel dilate*. Der in Formel 3-1 verwendete Korrekturfaktor wird aus dem Vergleich der Voxelzahl in Tabelle 16 bzw. der daraus berechneten Strukturoberfläche ermittelt, wodurch sich ein gemittelter Korrekturfaktor von rund 1,921 ergibt.

Tabelle 16: Vergleich der Ergebnisse aus der numerischen Ermittlung der volumenbezogenen, spez. Oberfläche für unterschiedliche Packungsporositäten und die Voxelanzahl an der Strukturoberfläche

Volumenspezifische Materialoberfläche $a_{V,S}$ [m^2/m^3]					
Packungsporosität ϵ [%]	23,6	28,2	33,3	40,1	45,3
Sum of Voxelfaces	5.190	5.627	5.762	5.460	5.256
Surface fit	3.414	3.743	3.794	3.654	3.427
Voxel dilate (ohne Korrekturfaktor)	3.295	4.097	4.526	4.958	5.095
Voxelanzahl an der Oberfläche					
Sum of Voxelfaces	783.716	849.692	870.136	824.464	793.803
Voxel dilate (ohne Korrekturfaktor)	380.088	444.354	455.780	448.797	421.086

3.2 Berechnungsmodelle für die Strömungssimulation

Für die numerische Herangehensweise zur Ermittlung strömungsrelevanter Parameter, wie den Druckverlust oder die Strömungsgeschwindigkeit, stehen in GeoDict verschiedene Berechnungsmodelle und Lösungsalgorithmen (kurz Solver genannt) zur Verfügung. Die Auswahl des geeigneten Modells hängt dabei von der Materialstruktur und der Strömungsgeschwindigkeit ab.

Im Falle eines sehr langsamen Flusses wird von einer laminaren oder auch zähen Strömung gesprochen, welche mit fortschreitender Geschwindigkeit nach einem Übergangphase in die turbulente Phase umschlägt. In einem laminaren Flusses bewegt sich das Fluid wie in Schichten, welche sich nicht vermischen. Ab einer bestimmten Grenze beginnen sich allerdings Quervermischungen auszubilden und eine turbulente Strömung entsteht. Diese Grenze wird nach [7] mithilfe der Reynoldszahl charakterisiert, einer dimensionslosen Kennzahl, welche die Trägheitskraft mit der Reibungskraft in Verhältnis setzt. Formel 3-3 zeigt die notwendigen Parameter für die Berechnung und in welcher Beziehung diese stehen. Überwiegt also die Trägheit die Reibung ab einem gewissen Punkt, so beginnen sich zunehmend Turbulenzen auszubilden.

$$Re = \frac{x * w}{\nu} = \frac{x * w * \rho_{fl}}{\eta_{fl}}$$

Formel 3-3

Die charakteristische Länge x stellt dabei das spezifische Längenmaß des betrachteten Strömungsproblems dar. Für die Berechnung der Reynoldszahl einer Strömung durch ein Rohr wird dessen Durchmesser herangezogen. In porösen Schichten, welche durch polydisperse Partikel aufgebaut sind, gibt es mehrere Möglichkeiten. Für die nachfolgenden Berechnungen wird als charakteristische Partikelgröße der harmonische Durchmesser d_{harm} herangezogen, welcher laut [5] sich besonders bei Korngrößenverteilungen eignet, bei welchen die Oberfläche wichtig ist. Die Berechnung erfolgt nach Formel 3-4 [5], wobei der darin befindliche Parameter d_{mi} für die mittleren Korngröße jeder Kornklasse steht.

$$d_{harm} = 100 * \frac{1}{\sum_i \left(\frac{\Delta Q_i}{d_{mi}} \right)}$$

Formel 3-4

Alternativ bietet sich auch nur der Sauterdurchmesser als charakteristische Länge an. Für geringe Reynoldszahlen und somit für eine zähe Strömung sind bei gegebenen Längenmaß x , eine langsame Strömungsgeschwindigkeit w und eine hohe kinematische Zähigkeit ν nötig. Die kinematischen Zähigkeit von Luft, welche als durchströmendes Fluid verwendet wird, beträgt im trockenen Zustand $157,9 * 10^{-7} \text{ m}^2/\text{s}$ bei einem Druck von 1 bar und einer Temperatur von 25°C. Aufgrund dieser Tatsache, sowie einer relativ hohen Gasgeschwindigkeit ($w \sim 1 \text{ m/s}$) wird eine turbulente Strömung erwartet.

Nach [10] werden für turbulente Strömungsverhältnisse und einer Schüttung aus kompakten Feststoffen die Navier-Stokes-Gleichungen als Grundmodell empfohlen. Dabei handelt es sich um ein System aus Differentialgleichungen zweiter Ordnung, welche laut [7] die Energie- und Impulserhaltung beschreiben. Eine analytische Lösung dieser Durchströmungsgleichungen ist aufgrund ihrer Komplexität allerdings unmöglich. Nur durch bestimmte Annahmen und Vereinfachungen kann durch Approximation ein Ergebnis für solch komplizierte Strukturen, wie sie hier vorliegen, gefunden werden. Nach [10] wird ein stationärer Zustand der Strömung angenommen, das heißt dass sich die Fließgeschwindigkeit zeitlich nicht ändert. Des Weiteren wird bei dem durchströmenden Medium von einem inkompressiblen, newtonischen Fluid ausgegangen. Gase besitzen eigentlich die Eigenschaft der Kompressibilität, womit ihre Dichte veränderlich wäre. Unter der Bedingung einer Machzahl unter 0,3 kann allerdings das Medium näherungsweise als inkompressibel behandelt werden; [7]. Diese Kennzahl berechnet sich aus dem Verhältnis der Fließgeschwindigkeit zur Schallgeschwindigkeit (siehe Formel 3-5). Die maximale durchschnittliche Strömungsgeschwindigkeit beträgt bei der nachfolgenden Simulation nur 1,6 m/s und bei einer Schallgeschwindigkeit von 346,3 m/s in der trockenen Luft bei 25°C und 1 bar [24] liegt die Machzahl weit unter dem Grenzwert ($Ma = 0,0046$). Als newtonisches Fluid wiederum werden Medien mit konstanter Viskosität bezeichnet. Die getroffenen Annahmen und Vereinfachungen sind somit unter den vorgegebenen Bedingungen zulässig.

$$Ma = \frac{w}{c_s}$$

Formel 3-5

$$Ma = \frac{1,6}{346,3} = 0,0046$$

Die Navier-Stokes-Gleichungen können somit in der vereinfachten Form (stationär, inkompressibel) über Formel 3-6 bis 3-8 angeschrieben werden, welche aus [7] entnommen sind. Die erste Zeile stellt die Kontinuitätsgleichung dar, welche die Erhaltung der Masse beschreibt. Die drei nachfolgenden Zeilen stellen die Impulsgleichungen für einen dreidimensionalen Raum dar. Wie bereits erwähnt spielt bei einer turbulenten Strömung die Trägheit eine Rolle und muss somit in der Berechnung berücksichtigt werden. Dies geschieht mittels des grün eingefärbten Teils der Gleichung, welcher im Falle eines laminaren Flusses vereinfachend weggelassen werden könnten. Der Term f auf der rechten Seite steht laut [4] für den Quellterm, also Kräfte, welche auf das Fluid wirken (z.B. die Gravitationskraft).

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

Formel 3-6

$$-v \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + \frac{1}{\rho_{fl}} \frac{\partial p}{\partial x} + \left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) = f_x$$

Formel 3-7

$$-v \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + \frac{1}{\rho_{fl}} \frac{\partial p}{\partial y} + \left(u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) = f_y$$

Formel 3-8

$$-v \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + \frac{1}{\rho_{fl}} \frac{\partial p}{\partial z} + \left(u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right) = f_z$$

Formel 3-9

3.2.1 Randbedingungen und Eigenschaften des Lösungsalgorithmus

Um für einen bestimmten Anwendungsfall die Gleichungen lösen zu können und ein eindeutiges Ergebnis zu bekommen, sind Rand- und Anfangsbedingungen notwendig. Anfangsbedingungen stellen zeitliche Vorgaben dar, zum Beispiel, eine Startgeschwindigkeit zum Zeitpunkt Null. Da hierbei allerdings nur mit stationären Problemen gerechnet wird, kann diese Kategorie außer Acht gelassen werden. Randbedingungen wiederum geben laut [10] die Flussrichtung vor und im Kontakt mit einer festen Wand bzw. Strömungsberandung geben diese auch an, wie sich der Fluss dort verhält. Bei strömenden Medien, welche einen Feststoff vorbeifließen, ist eine der wichtigsten Vorgaben die sogenannte „No-slip“-Bedingung. Dies bedeutet, dass das Strömungsprofil zum Feststoff hin in einer sehr schmalen Grenzschicht abnimmt bis es den Wert Null erreicht. Abbildung 30 stellt das Strömungsprofil entlang einer ebenen Wand dar und wie es sich durch das Hindernis verändert. Aus diesem Grund kann für jeden Punkt an der Oberfläche der Struktur die Fließgeschwindigkeit in alle Richtungen mit Null angenommen werden.

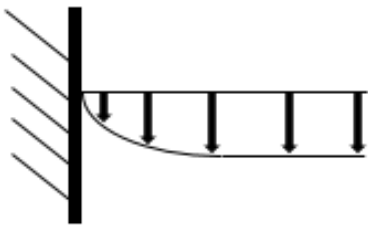


Abbildung 30: Verlauf des Strömungsprofils an einer ebenen Wand

Grundsätzlich kann bei den Simulationen entweder von der Strömungsgeschwindigkeit auf den Druckverlust oder umgekehrt geschlossen werden. Aus diesem Grund wird eine mittlere Fließgeschwindigkeit vorgegeben und anhand dieser andere strömungsrelevante Parameter hergeleitet. Für die Simulationen wird in einem Wertebereich zwischen 0,25 und 1,6 m/s für insgesamt fünf Geschwindigkeiten der Druckverlust ermittelt. Eine weitere Vorgabe ist die Fließrichtung, welche unabhängig von der augenblicklichen Ansicht der Struktur auf der Benutzeroberfläche immer in positive Z-Richtung verläuft. Sollte das dargestellte Element von einer anderen Seite durchflossen werden, so müssen die Achsen vertauscht werden. An den Seitenflächen (= Mantelflächen) des Berechnungsgebietes werden Symmetrie-

Randbedingungen angesetzt. Das heißt, es wird vereinfachend angenommen, dass sich die vorhandene Struktur über das Berechnungsgebiet hinaus in gespiegelter Weise in die Umgebung fortsetzt. In GeoDict wird dies als symmetrische „Boundary Condition in Tangential Directions“ angeführt. Des Weiteren wird für den Eintritt eine konstante Geschwindigkeit vorgegeben und am Austritt ein konstanter Druck.

Als weitere Randbedingung wird die sogenannte „Implicit region“ beim „Inflow“ und „Outflow“ hinzugefügt, also eine zusätzlich leere Voxelschicht vor und nach der eigentlichen Struktur in Z-Richtung. Der Lösungsalgorithmus approximiert das Ergebnis viel schneller, wenn die Voxelzahl in der Strömungsrichtung ein Ergebnis von 2^x ist mit $x \in \mathbb{N}$. Sollte dies nicht der Fall sein, kann mit Hilfe von „Implicit region“ einfach die fehlende Differenz zwischen der eigentlichen Höhe und den dementsprechenden Sollwert ausgeglichen werden. Diese Zusatzschicht hat keine Auswirkungen auf das Endergebnis und wird darin nicht berücksichtigt. Bei der ersten Modellpackung (Kugelpackung) beträgt die Höhe 500 Voxel, wodurch noch 12 Voxel bis 512 ($= 2^9$) fehlen. Diese Differenz wird durch ein Hinzufügen von 4 Voxel im Eintritts- und 8 Voxel im Austrittsbereich ausgeglichen. Die nächst höhere Potenz 2^{10} ergibt 1024, die Voxelzahl der beiden anderen Packungen in Z-Richtung hat allerdings nur 700. Die fehlenden 324 Voxel werden deshalb folglich aufgeteilt: Inflow: 108 Voxel; Outflow: 216 Voxel. Das Verhältnis 1:2 bleibt somit gleich, wie bei der Kugelpackung. In diesem Zusammenhang muss die sogenannte „VinPout“-Bedingung gewählt werden, welche nach [10] einen konstanten Eintrittsstrom und einen konstanten Druck am Austritt vorgibt.

Um das Gleichungssystem mit den vorgegebenen Randbedingungen lösen zu können, muss ein geeigneter Solver ausgewählt werden, welcher durch Approximation die Lösung numerisch annähert. In [10] wird der sogenannte „SimpleFFT-Solver“ („Simple Fast Fourier Transform“) für die Navier-Stokes-Gleichungen empfohlen, insbesondere wenn die zu durchströmende Struktur eine niedrige Porosität aufweist. Als Kriterium, welches den Solver beendet, wird eine Toleranz angegeben. Sobald die Werte für die Permeabilität während der Näherung keiner großen Schwankung mehr unterliegen und beginnen stationär zu werden, stoppt der Solver. Als Toleranz wird ein Wert von 0,01 eingegeben. Wichtig ist bei dieser Art von Solver des Weiteren die Angabe der Relaxationsfaktoren „Pressure relaxation“ und „Velocity relaxation“. Diese können zwischen den Extremen „Stable“ = 0 und „Fast“ = 1 variiert werden und beeinflussen das Konvergenzverhalten des Algorithmus. Nach [10] führt ein niedrigerer Wert zu mehr Iterationen, damit zu einer längeren Berechnungszeit und wird für höhere Reynoldszahlen empfohlen. Bei „Fast“ würde die Zeit erheblich gekürzt werden, allerdings besteht das Risiko, dass der Solver nicht konvergiert. In allen Simulationsvorgängen wird deshalb für „Velocity relaxation“ ein Wert von 0,5 ausgewählt und für „Pressure relaxation“ eine etwas schnellere Rechenzeit mit 0,8.

In der nachfolgenden Tabelle sind nochmals alle Randbedingungen, sowie weiteren Angaben für den Solver und Eigenschaften des Fluids zusammengefasst.

Tabelle 17: Zusammenfassung aller Werte für die Parameter, welche in dem Modul FlowDict der Software GeoDict eingegeben werden

Eingabewerte für FlowDict			
Stoffdaten des Fluid ¹⁾			
Fluid	Luft		
Temperatur [°C]	25		
Dichte [kg/m ³] ¹⁾	1,184		
Dyn. Viskosität [Pa·s] ¹⁾	1,8515*10 ⁽⁻⁵⁾		
Randbedingungen			
Strömungsrichtung	+Z-Achse		
VinPout	Ja		
Mittlere Geschwindigkeit [m/s]	[0,25; 0,6; 1; 1,3; 1,6]		
Tangentiale Richtung	symmetrisch		
"No-Slip"	Ja		
Höhe [Voxel]		500	700
Add Implicit Region	Inflow	4	108
	Outflow	8	216
Eigenschaften des Solvers			
Solver-Typ	SimpleFFT		
Stopp-Kriterium	Toleranz	0,01	
Velocity relaxation	0,5		
Pressure relaxation	0,8		
Charakteristische Länge [m] ²⁾	0,00167		
	0,00278		

1) Die angegebene Dichte und dynamische Viskosität der Luft werden in GeoDict angezeigt

2) Bei den angegebenen charakteristischen Längen handelt es sich um den harmonischen Durchmesser der Korngrößenverteilung aus [5] und darunter den harmonische Durchmesser berechnet aus der Korngrößenverteilung aus [25]

3.3 Empirische Druckverlustgleichungen und Permeabilität

Den aus den Simulationen ermittelten Druckverlustkurven werden Gleichungen, welche den Druckverlust beschreiben, gegenübergestellt. Bei diesen steht der spezifische Druckverlust nicht nur über physikalische Kenngrößen, sondern auch über strukturelle Parameter mit der Fließgeschwindigkeit in Verbindung. Dadurch kann bei einer Übereinstimmung des numerischen Modells und einer der Gleichungen der Einfluss der Parameter und ihre Auswirkungen bei einer Schwankung abgeschätzt werden. Bei den nachfolgend beschriebenen Modellen handelt es sich um empirisch ermittelte Gleichungen, das heißt dass diese durch Experimente entstanden und dadurch auch nur in gewissen Bereichen und unter bestimmten Bedingungen gültig sind.

Eine der empirischen Grundgleichungen zur Beschreibung der Durchströmung einer porösen Schicht und des daraus entstehenden Druckabfalls ist die Darcy-Gleichung, welche diesen mit der Strömungsgeschwindigkeit proportional in Verbindung setzt (siehe Formel 3-10). Der Gültigkeitsbereich erstreckt sich allerdings nur bis zu einer Reynoldszahl $\leq \sim 3$, wobei diese sich hierbei aus der mittleren Leerrohrgeschwindigkeit und der mittleren Partikelgröße als charakteristische Länge ergibt. Dies bedeutet, die Gleichung darf nur für zähe Strömungen eingesetzt werden.

$$\frac{\Delta p}{H} = \frac{\eta}{B} * w$$

Formel 3-10

Die strukturellen Eigenschaften der zu durchströmenden Schicht sind in dem Proportionalitätsfaktor B zusammengefasst, welcher Durchlässigkeit, Permeabilität oder Durchgasbarkeit genannt wird. Diese ist definiert als das Verhältnis aus der mittleren Partikelgröße zum Quadrat und einer Konstante, welche von der Porosität ε abhängt, dargestellt in Formel 3-11. In dieser Definition wird seine Einheit in m^2 angegeben, wobei eine weitere Möglichkeit zur Angabe die Einheit darcy darstellt. Ein darcy sind laut [24] $9,869233 * 10^{-13} \text{ m}^2$. Da sich im laminaren Fall der Druckverlust mit der Geschwindigkeit linear verändert, ergibt sich in diesem Bereich eine konstante Permeabilität. Damit ist diese eine rein von Material abhängige Konstante. In der turbulenten Strömung verändert sich allerdings der Druckverlust nicht mehr linear zur Strömungsgeschwindigkeit. Die Permeabilitätsgröße B wird damit von Druckveränderungen, als auch der Viskosität beeinflusst und stellt damit laut [10] keine reine Materialgröße mehr dar.

$$B \equiv \frac{\bar{x}^2}{K(\varepsilon)}$$

Formel 3-11

Da die Permeabilität allerdings den Porenquerschnitt charakterisiert, muss sie für jede Schicht empirisch ermittelt werden. In [5] wurde der Parameter mit einem Permeameter gemessen.

Bei dem Vorgang wird die zu untersuchende Probe in einen zylindrischen Behälter gefüllt und der Druckverlust bei der Durchströmung mit Luft gemessen. Die Permeabilität ergibt sich schließlich aus Formel 3-12 und wird in JPU (Japanese Permeability Unit) angegeben, welche eine international verbreitete Einheit darstellt. Dabei ist jedoch besonders auf die Einheiten der in der Formel enthaltenen Parameter zu achten. Der Luftvolumenstrom \dot{V} muss in Nm^3/min , die Querschnittsfläche in m^2 und Höhe der Schüttung in mm , sowie Druckverlust über die Schüttung in $\text{mm H}_2\text{O}$ eingesetzt werden.

$$JPU = \frac{\dot{V}}{A} * \left(\frac{H}{\Delta p}\right)^{0,6}$$

Formel 3-12

Neben den Einheiten m^2 und JPU existieren noch weitere mögliche Einheiten, wie zum Beispiel m/s . Die Permeabilität wird in Bezug auf die Sinterqualität und –produktivität in vielen wissenschaftlichen Artikeln als Schlüsselfaktor gesehen.

Eine weitere Grundgleichung stellt die Carman-Kozeny-Gleichung dar, bei welcher ebenfalls der spezifische Druckverlust $\frac{\Delta p}{H}$ proportional zur Geschwindigkeit w ist. Ausgangspunkt der Carman-Kozeny-Gleichung stellt die Hagen-Poiseuille-Gleichung für einen laminaren Fluss durch einen kreisförmigen Kanal dar (Formel 3-12), wobei die Herleitung aus [20] entnommen ist.

$$\frac{\Delta p}{H} = \frac{32 * \eta * w}{d^2}$$

Formel 3-13

Es wird nun angenommen, dass das Schüttgut aus vielen verwundenen Kapillaren besteht, welche alle einen äquivalenten Durchmesser D_e besitzen mit einer äquivalenten Höhe H_e , durch welche das Medium mit der tatsächlichen Geschwindigkeit w_e fließt. Abbildung 31 illustriert die Modellvorstellung. In Formel 3-13 eingesetzt ergibt sich die neue Formel 3-14. Die tatsächlichen Geschwindigkeit wird mittels der Porosität in die Leerrohrgeschwindigkeit w umgerechnet (Formel 3-15) und die tatsächliche Länge der Kapillaren mittels eines Proportionalitätsfaktors mit der Schüttguthöhe in Verbindung gebracht (Formel 3-16). Der Äquivalentdurchmesser D_e wird definiert als Vierfaches des durchflossenen Querschnittes dividiert durch den benetzten Umfang. Die durchflossene Querschnittsfläche kann einfach über die Porosität mit dem gesamten Querschnitt des Schüttgutes in Beziehung gebracht werden. Die Ermittlung des benetzten Umfangs erfolgt durch das Verhältnis der gesamten, benetzten Oberfläche und der Betthöhe, wobei der in Formel 3-17 abgebildete Parameter $a_{V_{ges,s}}$ die Partikeloberfläche bezogen auf das gesamte Bettvolumen darstellt. Dieser lässt sich auch in Abhängigkeit der bereits bekannten spezifischen Oberfläche $a_{V,s}$ darstellen. Durch das Substituieren von der in Formel 3-14 dargestellten Parameter durch die Formel 3-15, Formel

3-16 und Formel 3-18 ergibt sich schließlich die Carman-Kozeny-Gleichung, dargestellt in Formel 3-19.

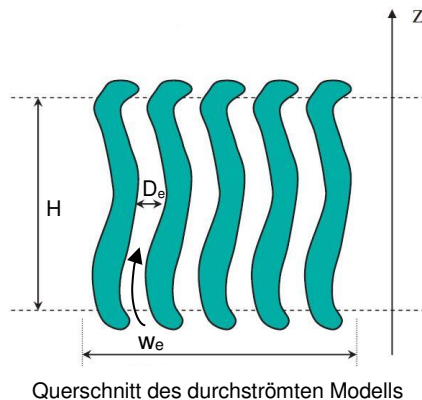


Abbildung 31: Modellvorstellung von Carman-Kozeny im Querschnitt [17]

$$\frac{\Delta p}{H_e} = K_1 * \frac{\eta * w_e}{D_e^2}$$

Formel 3-14

$$w_e = \frac{w}{\varepsilon}$$

Formel 3-15

$$H_e = K_2 * H$$

Formel 3-16

$$\begin{aligned} \text{benetzter Umfang} &= \frac{\text{ges. benetzte Oberfläche}}{\text{Höhe}} = \frac{a_{V_{ges,S}} * A * H}{H} \\ &= a_{V,S} * (1 - \varepsilon) * A \end{aligned}$$

Formel 3-17

$$D_e = \frac{4 * \text{durchflossene Querschnittsfläche}}{\text{benetzter Umfang}} = \frac{4 * \varepsilon * A}{a_{V,S} * (1 - \varepsilon) * A}$$

Formel 3-18

$$\frac{\Delta p}{H} = K_3 * \frac{(1 - \varepsilon)^2}{\varepsilon^3} \eta * w * a_{V,S}^2$$

Formel 3-19

Im Gegensatz zur Darcy-Gleichung werden nicht die Porosität und die charakteristische Partikelgröße in der Permeabilität zusammengefasst, sondern direkt in der Formel angegeben. Dadurch ist der Einfluss der Porosität besser bestimmbar. Nach [8] kann für die Konstante $K_{(3)}$ in einem Bereich zwischen $0,3 \leq \varepsilon \leq 0,65$ einfach der Faktor 4 verwendet werden. Andere Quellen ([20], [23]) geben wiederum eine Konstante von 5 an, wobei [23] diesen Wert für niedrige Porositäten und einer Partikelform, welche annähernd einer perfekten Kugel nahe kommt, empfiehlt. Als Einflussgrößen werden die Porosität, Partikelform und -größe angegeben, wobei eine detaillierte Erklärung über deren Gewichtung und Auswirkungen fehlen. Aus diesem Grund wird einfach der Mittelwert aus den vorgeschlagenen Koeffizienten

verwendet und ein Wert von 4,5 angenommen. Die spezifische Oberfläche nimmt eine dominante Stellung ein, da diese mit dem Quadrat in die Formel eingeht. Durch die heterogenen Ausgangsstoffe ändert sich der Wert kontinuierlich und kleine Abweichungen in der Bestimmung führen zu großen Auswirkungen im Endergebnis. In Tabelle 3 sind durch die unterschiedlichen Analysemethoden alleine drei verschiedene Ergebnisse entstanden. Im direkten Vergleich der Werte 45,98 und 50,16 cm²/cm³ zeigt sich eine geringe, relative Differenz von 8,3 %. In die richtige Einheit umgerechnet (m²/m³) und quadriert weichen die Ergebnisse bereits rund 16 % voneinander ab.

Die angegebene Gleichung (Formel 3-19) darf allerdings nur für zähe Strömungen eingesetzt werden. Wie bereits erwähnt ist im Falle von Turbulenzen der Druckabfall nicht mehr linear zur Durchströmungsgeschwindigkeit. Diese Nicht-Linearität kann allerdings durch einen zusätzlichen Term in der Gleichung berücksichtigt werden, welcher zu der bestehenden addiert wird und stellt somit den turbulenten Anteil in der Formel dar. Die dort angeführte Konstante variiert von jener des laminaren Anteils und kann ebenfalls aus Experimenten ermittelt werden. Das Werk [23] gibt als einer von wenigen einen Wert für den turbulenten Koeffizienten mit 0,292 an, womit auch Durchströmungen mit höheren Reynoldszahlen berechnet werden können. In der nachstehenden Formel 3-20 ist die vollständige Version angeschrieben.

$$\frac{\Delta p}{H} = 4,5 * \frac{(1 - \varepsilon)^2}{\varepsilon^3} * a_{V,S}^2 * \eta_g * \bar{w} + 0,292 * \frac{1 - \varepsilon}{\varepsilon^3} * a_{V,S} * \rho_g * \bar{w}^2$$

Formel 3-20

Bekannter ist die Formel allerdings in Abhängigkeit des Sauterdurchmessers anstatt der spezifischen Oberfläche unter dem Namen Ergun-Gleichung (siehe Formel 3-21). Erguns Modell basiert somit auf denselben Grundannahmen wie Carman-Kozeny, ist aufgrund der Erweiterung allerdings auch für einen größeren Reynoldsbereich einsetzbar. Nach [22] ist sein Anwendungsgebiet Schüttungen aus gebrochenem Mahlgut, wobei explizit die Materialien Koks, Erz und Steine angeführt sind. Diese sollten eine relativ enge Korngrößenverteilung besitzen und deren Morphologie, sowie Porosität gut bekannt sein, da diese sich stark auf das Ergebnis auswirken. Als Grenzen der Reynoldszahl ist ein Bereich zwischen $3 \leq Re \leq 10\,000$ angegeben, wobei mit steigender Geschwindigkeit der turbulente (rechte) Term der Gleichung überwiegt. Die Konstanten dieser empirischen Gleichung werden meistens mit 150 für den laminaren und 1,75 für den turbulenten Teil angegeben; [20, 22]. In seiner Arbeit schlägt [16] allerdings stattdessen die Werte 323 und 3,78 vor, welche aufgrund der Übereinstimmung des untersuchten Materials auch in weiterer Folge übernommen werden.

$$\frac{\Delta p}{H} = 323 * \frac{\eta_g * (1 - \varepsilon)^2}{d_{32}^2 * \varepsilon^3} * \bar{w} + 3,78 * \frac{\rho_g * (1 - \varepsilon)}{d_{32} * \varepsilon^3} * \bar{w}^2$$

Formel 3-21

4 Diskussion der erzeugten Packungsstrukturen

In diesem Kapitel werden die virtuell erstellten Modellpackungen diskutiert, sowie die Korngrößenverteilungen auf denen diese basieren. Auch eine Diskussion der verschiedenen Analysemethoden zur Ermittlung der spezifischen Oberfläche bzw. des Sauterdurchmessers findet sich auf den nachfolgenden Seiten. Neben den angewendeten Methoden werden auch experimentelle Untersuchungsvarianten herangezogen. Des Weiteren findet sich eine Diskussion der erstellten Modelle hinsichtlich ihrem Erscheinungsbild und den geforderten Ansprüchen.

4.1 Vergleich der Korngrößenverteilungen und Strukturparameter

Als Grundlage für die virtuellen Modellpackungen dienen Siebanalysen, welche teils händisch, teils photooptisch und computerunterstützt durchgeführt wurden. Erstere zeichnet sich insbesondere durch seine einfache Handhabung aus, sodass auch das Analysieren von feuchten Proben kein Problem darstellt. Allerdings entstehen durch die vielen Handgriffe, dem manuellen Abwiegen usw. während der Siebung viele Fehlerquellen, welche das Ergebnis völlig verzerren und damit unbrauchbar machen könnten. Des Weiteren können viel weniger Kornklassen ermittelt werden, als bei einer computerunterstützten Auswertung. Für die photooptische Analyse der Korngrößenverteilung, welche in [5] angewendet wurde, wurde die Probe allerdings vorher getrocknet. Dadurch erhöht sich der Feinanteil und führt damit zu einer falschen Basis für ein weiterfolgendes Simulationsmodell einer nativen Grünmix-Packung. Abbildung 32 zeigt vergleichend alle Korngrößenverteilungen aus dem Kapitel 2.1.2 und 2.3.1 in Form ihres relativen Durchgangs ΔQ . Für eine bessere Vergleichbarkeit wird die computerunterstützte Verteilung (gelbes Histogramm) von 14 auf 8 Kornklassen reduziert und damit der händischen Siebanalyse der Darstellung nach angepasst. Nur bei Kornklassen $<0,5$ mm driften die Grenzen etwas auseinander. Die Summe des relativen Durchgangs für den Feinanteil (<1 mm) des gelben Histogramms ergibt 13,23 % und liegt damit nur knapp unter jener Verteilung, dessen Probe erst nach Monaten analysiert wurde ($\sum \Delta Q(x < 1 \text{ mm}) = 14,49$ %). Mit einem Anteil von gerade einmal 2,18 % besitzt das rote Histogramm, welches als Grundlage des „Polyeder-36“-Modells Verwendung findet, die geringste Menge an Kleinstkörnern. Der Grund liegt, wie bereits auf S. 30 erwähnt, in der enthaltenen Feuchtigkeit, welche während der Granulation zugegeben wird um eine Agglomeration der feinen und größeren Partikel hervorzurufen. Die Proben werden für die Lagerung in geschlossene Plastikbehälter bei Raumtemperatur aufbewahrt, wodurch das Wasser verdunstet und sich durch Kondensation bevorzugt am Deckel sammelt. Dadurch fehlt das notwendige Bindemittel und der Feinanteil erhöht sich während der späteren Siebanalyse. Der Zeitpunkt der Analyse nach der Probeentnahme spielt somit eine bedeutende Rolle in Bezug auf die resultierende Korngrößenverteilung. Im Falle der Korngrößenverteilung aus [5] wird angegeben, dass diese vor der Analyse im Trockenschrank getrocknet wurde, wodurch sich die hohen Werte für den Kornklassenbereich <1 mm ergeben können.

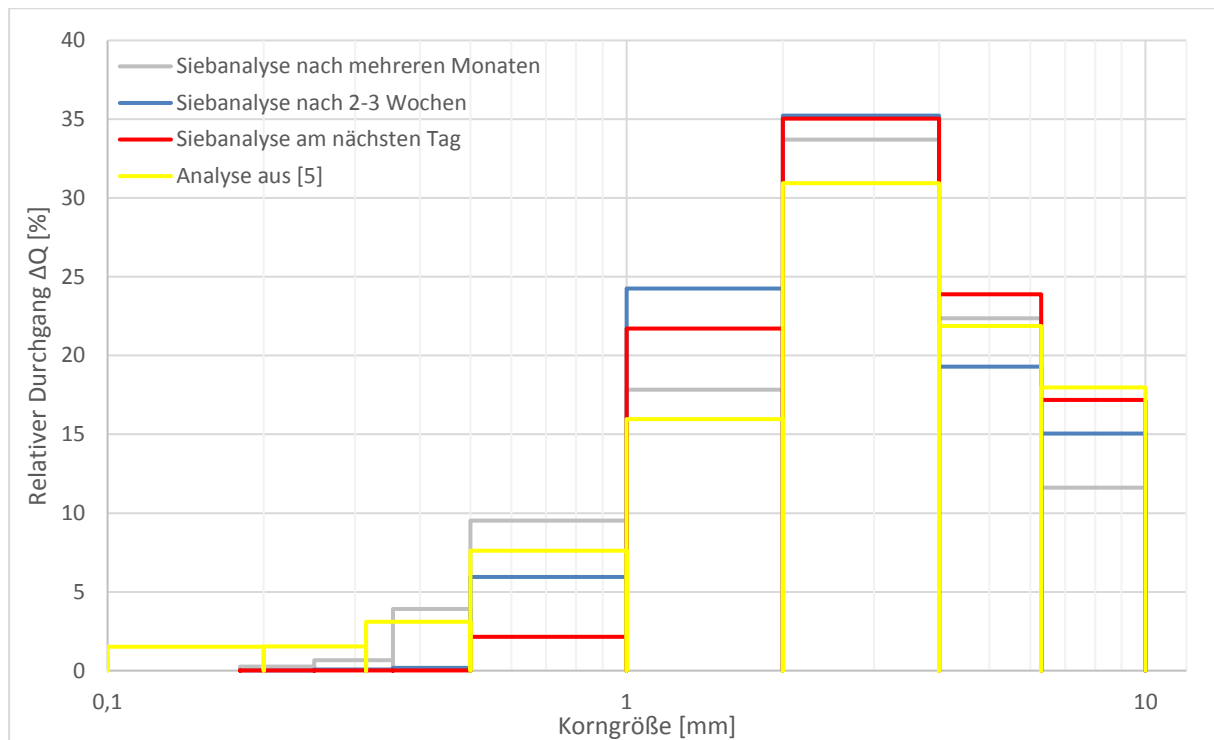


Abbildung 32: Vergleichende Darstellung der relativen Durchgänge ΔQ von Korngrößenverteilungen aus [5] und [25]

Die Werte der Korngrößenverteilung haben auch große Auswirkungen auf die spezifische Oberfläche bzw. dem Sauterdurchmesser, welche weiterführend in empirische Druckverlustgleichungen Anwendung finden. Aus diesem Grund sind unterschiedliche Analysemethoden von Vorteil, um ein Ergebnis zu bestätigen, zu verwerfen oder zumindest den Wertebereich eingrenzen zu können. Im Allgemeinen können diese Methoden in theoretische und praktische Ermittlungsvarianten aufgeteilt werden. Zu den ersteren zählt, z.B. die Anwendung der Verteilungsfunktion RRSB, woraus jene Parameter (n , dp') zur Berechnung der spezifischen Oberfläche ermittelt werden können. Diese Methode ist den Kapiteln 2.1.3 bis 2.1.5 detailliert beschrieben. Vorteil ist die schnelle und einfache Berechnungsmöglichkeit, wodurch innerhalb kurzer Zeit ein Wert für die spezifische Oberfläche und den Sauterdurchmesser angeschrieben werden kann. Wie bereits oben erwähnt, kann es allerdings leicht zu einer verfälschten Korngrößenverteilung kommen, welche wiederum die Feinheitparameter beeinflusst. Wird des Weiteren eine schlechte Wahl für die Bezugswerte von n und dp' herangezogen, kann das Ergebnis noch stärker abweichen. Nicht zu unterschätzen ist auch der Einfluss des Formfaktors, welcher oftmals nur geschätzt ist, aber direkt proportional in die spezifische Oberfläche eingeht. Damit bestehen viele, eventuelle Fehlermöglichkeiten, welche zum Schluss das Ergebnis stark verfälschen können. Dasselbe trifft auf die Anwendung des RRSB-Diagramms zu, durch welches eine grafische Auswertung der Korngrößenverteilung möglich ist. Hier kann an einem Randmaßstab die spezifische Oberfläche ermittelt werden, wobei für die genaue Vorgehensweise auf den letzten Absatz des Kapitels 2.1.5 verwiesen wird. Trotz der wiederum schnellen Ermittlungsmöglichkeit, können in diesem Fall viele Ungenauigkeiten zu einer starken Abweichung vom richtigen Wert für die spezifische Oberfläche führen. Da es sich um ein

doppelt logarithmisches Diagramm handelt, ist das genaue Eintragen von Werten eventuell fehlerbehaftet oder nur ungenau möglich. Dazu kommt nach fertigen Markieren aller Punkte das approximative Einzeichnen einer geraden Linie, welche möglichst alle Punkte annähert. Auch das Ablesen am Randmaßstab ist ein Schätzungsprozess mit Fehlermöglichkeiten. Eine direkte grafische Ermittlung des Sauterdurchmessers gibt es nicht, er muss in diesem Fall über Formel 2-13 berechnet werden.

Diesen Beispielen für theoretische Analysemöglichkeiten der spezifischen Oberfläche bzw. des Sauterdurchmesser seien praktische Versuchsdurchführungen gegenübergestellt. Zu diesen zählt etwa die Blaine-Methode, bei welcher die Grundlage des Experiments die Carman-Kozeny-Gleichung (Formel 3-19) ist; [22]. Während der Versuchsdurchführung wird dabei Unterdruck angelegt und ein bestimmtes Volumen an Luft durch eine geschüttete Probe gesaugt. Die dafür benötigte Zeitspanne wird gemessen und in die nachfolgende Formel 4-1 eingesetzt. Die Konstante K_{Bl} stellt dabei eine gerätespezifische Konstante dar. Nach [22] ist der Messbereich für die spezifische Oberfläche allerdings nach oben begrenzt mit $14.000 \text{ cm}^2/\text{cm}^3$.

$$a_{V,s} = K_{Bl} * \frac{\varepsilon^3}{(1 - \varepsilon^2)} * \frac{\Delta t}{\eta_g}$$

Formel 4-1

Einen anderen Zugang im Gegensatz zu dem Durchströmungsverfahren bietet die Ermittlung der spezifischen Oberfläche über Gasadsorptionsverfahren. Nach [1] können über die BET-Methode und CT-Scans wichtige Strukturparameter, wie die Porosität und spezifische Oberfläche, bestimmt werden. Bei der BET-Methode wird der Effekt ausgenutzt, dass sich Gasmoleküle an der Oberfläche anhaften, wobei mehr Moleküle adsorbieren, desto höher der Druck ist.

Die BET-Methode ist im Gegensatz zur Blaine-Methode bezüglich der Messgrenze nicht nach oben beschränkt. Ein Unterschied zur Blaine-Methode ist, dass bei letzterer nur die äußere Oberfläche ermittelt wird. Durch die Gasadsorption kann allerdings auch die innere Oberfläche und somit die gesamte Oberfläche ermittelt werden. Die Ergebnisse aus der BET-Methode ergeben laut [1] einen Wert von $1.165 \text{ m}^2/\text{g}$ bei einer Porosität von $32,18 \%$. Dieser ergibt nach der Umrechnung mittels der Rohdichte von $\rho_s = 3,29 \text{ g/cm}^3$ (aus [5] entnommen) eine spezifische Oberfläche von rund $3.8 \cdot 10^6 \text{ m}^2/\text{m}^3$. Bei der dafür verwendeten Probe handelt es sich allerdings um ein unreduziertes, fertiges Sinterstück. Sie ist also eher mit der spezifischen Oberfläche der modifizierten CT-Struktur vergleichbar, deren Porosität $33,3 \%$ beträgt (siehe Tabelle 16). Allerdings weicht der BET-Wert der spezifischen Oberfläche sehr stark von den numerischen Ergebnissen ab, welche erheblich kleiner sind. Eine weitere Möglichkeit bietet die photooptische Analyse, welche in [5] angewendet wurde um an die spezifische Oberfläche zu gelangen. Der Vorgang ist auf S.16 erklärt. Nachteilig ist dabei aber die vorangegangene Trocknung der Probe, welche in Folge dessen den Feinanteil erhöht und den Wert verfälscht.

4.2 Vergleich der Modellpackungen

Für jede der generierten Modellpackungen wurde die vorgegebene Porosität und Korngrößenverteilung erfüllt, wobei für die Modelle „Kugel“ und „Polyeder-46“ sogar dieselben Vorgaben galten. Dennoch sehen die beiden Packungen im Vergleich sehr unterschiedlich aus. Das Modell „Kugel“ zeigt viele Hohlräume und bei einem Scan durch die Packung ist das Bild mehr von größeren Partikeln, als kleineren geprägt. Im Zuge dessen ist ein eher niedriger Druckverlust bei einer Durchströmung zu erwarten. Insgesamt ist „Kugel“ ein schlechtes repräsentatives Modell für eine Schüttung aus den Sinter-Rohmaterialien. Das Modell „Polyeder-46“ wiederum zeigt eine viel kompaktere Schüttung mit einem hohen Anteil an Körnern < 1 mm. Es scheint auf den ersten Blick unglaublich, dass diese Packung dieselbe Porosität wie „Kugel“ besitzt. Diese feine Verteilung an kleinen Partikeln wird in den nachfolgenden Strömungssimulationen zu einem höheren Widerstand führen und den Druckverlust nach oben treiben. Wird nun auch das Modell „Polyeder-36“ zum Vergleich herangezogen, zeigt es äußerlich keinen großen Unterschied zum Modell „Polyeder-46“. Beide Packungen schauen von außen sehr kompakt aus, wobei bei genauerer Betrachtung der Ränder mehr Luftlöcher in „Polyeder-46“ zu sehen sind. Dass die größte Kornklasse der Modellpackung „Polyeder-46“ mehr als 2 mm größer ist, fällt nicht auf. Die letzten beiden Modelle stellen bereits gute, repräsentative Packungen einer realen Schüttung aus dem Sinter-Rohmaterial dar.

5 Vergleichende Diskussion der Durchströmungs-Charakteristik

Nachfolgend sind die verschiedenen Ergebnisse aus den numerischen Berechnungen angeführt, sowie die experimentellen Analysewerte zum Vergleich. Dabei wird ebenfalls deren Probenvorbereitung und Durchführung hinsichtlich ihrer Vergleichbarkeit analysiert. Unterschiedliche Faktoren, wie eine zu hohe Verdichtung des Schüttgutes in dem Analysenbehälter, eine zu trockene Probe oder ähnliches, können zu signifikanten Abweichungen in den Ergebnissen führen. Eine genauere Betrachtung der Experimente ist daher unerlässlich. Anschließend wird am Ende jedes Unterkapitels in einem Diagramm die aus den Simulationen entstehende Druckverlustcharakteristik mit empirischen Modellgleichungen, sowie mit den experimentellen Auswertungen verglichen. Durch mögliche Übereinstimmungen, aber auch Differenzen lassen sich Rückschlüsse auf die Simulationen und ihre Brauchbarkeit ziehen.

5.1 Ergebnisse aus der Strömungssimulation für Packung „Kugel“

Die experimentellen Ergebnisse, welche im Anschluss den Simulationen gegenübergestellt werden, entstammen aus [5]. Diese Arbeit stellt auch die Korngrößenverteilung bereit, anhand welcher die sphärische und mehreckige Modellpackung aufgebaut sind. In Kapitel 2.1.2 sind die genauen Werte für weitere Details aufgelistet. Die Probe mit der entsprechenden Verteilung wurde in einer Permeabilitätsanlage hinsichtlich der Durchgasbarkeit untersucht und dabei der spezifische Druckverlust in Abhängigkeit der Leergasdurchströmung gemessen. Die Probenvorbereitung umfasste zuerst eine Trocknungsphase, um einen bestimmten Feuchtegehalt später einstellen zu können, welcher sich hierbei auf 3,74 % beläuft. Die Zugabe des Wassers erfolgte dabei in einem Intensivmischer, auf welchen anschließend der Granulierprozess in einer Mischtrommel folgte. Die Permeabilitätsanlage besteht aus einem zylindrischen Gefäß (= Retorte), in welches die Probe eingefüllt wird und einem Messsystem zur Erfassung der benötigten Daten. Abbildung 33 gibt den genauen Aufbau der verwendeten Permeabilitätsanlage wieder. Die Retorte besitzt einen Innendurchmesser von 80 mm und es wurde vollständig bis zu einer Höhe von 124 mm mit dem Gemenge aufgefüllt. Während der Versuchsdurchführung wurde die Drehzahl des Gebläses erhöht und somit eine stetig steigende Durchströmungsgeschwindigkeit erzeugt. Die in der Abbildung unten dargestellte Messblende erfasste den genauen Volumenstrom und der Unterdruckmesser den durch die Strömung hervorgerufenen Druckverlust. Durch die Auflast von oben wurde in weiteren Schritten zusätzliches Gewicht auf die Schüttung ausgeübt, um die Dichte zu erhöhen und somit den Druckverlust in tieferen Schichten zu simulieren. Diese werden durch das Eigengewicht der darüber liegenden Partikeln immer mehr zusammengedrückt. Da sich die Körner in den erstellten Modellpackungen allerdings nur im besten Fall berühren und ansonsten quasi im Raum „schweben“, werden als Vergleichswerte nur jene ohne Auflast herangezogen. Luftgeschwindigkeit und der dazugehörige spezifische Druckverlust wurden im bereitgestellten

Datensatz in EXCEL bereits ermittelt und können dadurch einfach direkt übernommen werden. Diese Daten dienen als Vergleichswerte für die Druckverlustkurven.

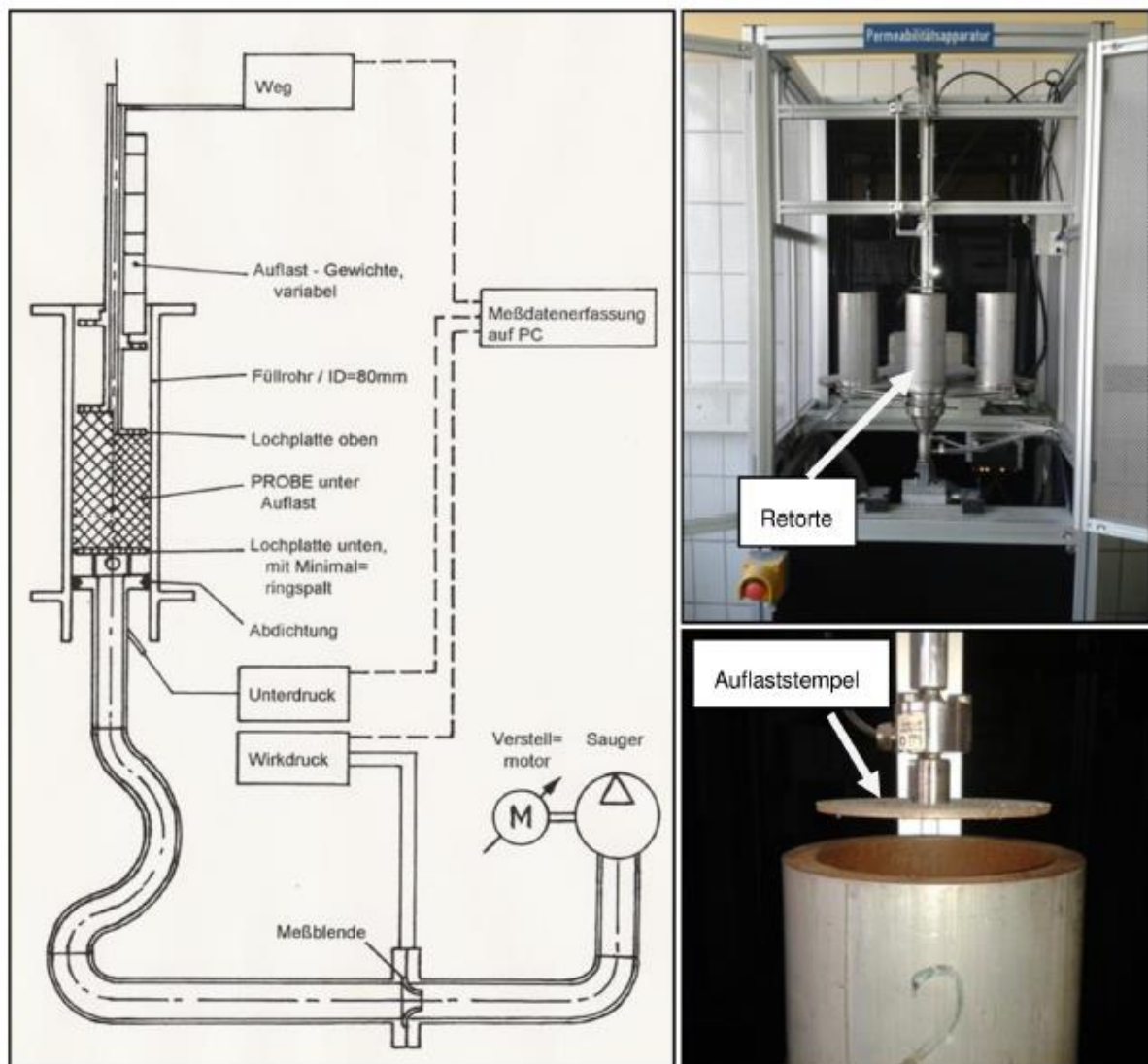


Abbildung 33: Aufbau der Permeabilitätsanlage [5]

Des Weiteren werden für die durchströmte Schüttung deren Schüttdichte, die mittlere Rohdichte und die Porosität angegeben. Erstere berechnete sich aus der eingewogenen Masse und dem Schüttvolumen, welches aufgrund der Maße der Retorte bekannt ist ($\rho_{schütt} = 1,767 \text{ g/cm}^3$). Die mittlere Rohdichte ρ_p der Partikel wurde in [5] aufwendig über die Massenanteile der Einzelkomponenten und deren Rohdichten ermittelt, wodurch sich ein Wert von $3,29 \text{ g/cm}^3$ ergibt. Beide Ergebnisse liefern die Eingabewerte für die Porosität, deren Berechnung einfach über Formel 5-1 erfolgt. Das Resultat ($\epsilon = 46,3 \%$) ist auch die Grundlage der Packungsmodelle von „Kugel“ und „Polyeder-46“.

$$\varepsilon = 1 - \frac{\rho_{schütt}}{\rho_s} = 1 - \frac{1,77}{3,29} = 0,463$$

Formel 5-1

Für die Erstellung der charakteristischen Druckverlustkurven wird, wie bereits erwähnt, für die Simulationen eine bestimmte mittlere Geschwindigkeit vorgegeben und der daraus entstehende spezifische Druckverlust numerisch ermittelt. Nebenbei berechnet das Programm allerdings viele weitere Parameter, welche als Vergleichsdaten herangezogen werden können. Aus dem angegebenen harmonischen Durchmesser als charakteristische Partikelgröße wird, zum Beispiel, die Reynoldszahl näherungsweise berechnet. Der Parameter wird in [5] nicht direkt angegeben, kann aber dank der im Datensatz beigelegten Werte für die Stoffparameter und der gleichen charakteristischen Partikelgröße einfach über Formel 3-3 ermittelt werden. Für die Luftdichte ρ_L wird ein Wert von $1,1839 \text{ kg/m}^3$ in der EXCEL-Tabelle angegeben und als dynamische Viskosität $\eta_L = 1,82 \cdot 10^{-5} \text{ Pas}$. Durch lineare Interpolation mit Stoffwerten der Luftdichte aus [24] entspricht dies einer Temperatur von rund 21°C .

Bei der numerischen Simulation wird zwar ebenfalls die Permeabilität in Z-Richtung berechnet, ein direkter Vergleich mit den experimentellen Ergebnissen ist allerdings erst nach Umrechnung der Einheiten möglich. Der Grund liegt in den unterschiedlichen Einheiten für die Permeabilität, welche in [5] in m/s und in GeoDict in m^2 angezeigt wird. Eine Neuberechnung der numerischen Permeabilität in JPU erscheint am sinnvollsten, da diese Größe in Experimenten und auch international Verwendung findet. Die Berechnung ergibt sich aus Formel 3-12, wobei einige Umrechnungen nötig sind. Der Volumenstrom muss in Nm^3/min umgewandelt werden, welches über die nachfolgende Formel 5-2 erfolgt. Die Normtemperatur T_N entspricht dabei einem Wert von $273,15 \text{ K}$. Die Temperatur bei der Durchführung der Simulationen beträgt 25°C , welche allerdings nicht für die Experimente verwendet wird. Stattdessen wird die errechnete Temperatur mit 21°C herangezogen, welche am Ende des vorangegangenen Absatzes erwähnt ist.

$$\dot{V} \left[\frac{\text{Nm}^3}{\text{min}} \right] = w * A * 60 * \frac{T_N}{T} = w * A * 60 * \frac{273,15}{273,15 + 25}$$

Formel 5-2

Des Weiteren ist eine Umwandlung des Druckverlustes erforderlich, welcher für die Einheit JPU in $\text{mm H}_2\text{O}$ übertragen werden muss. Der Umrechnungsfaktor ist aus [24] entnommen und wird dort mit $1 \text{ mm H}_2\text{O} = 9,80665 \text{ Pa}$ angegeben. Wird nun Formel 5-2 in Formel 3-12 eingesetzt, sowie der Umrechnungsfaktor für den Druckverlust ergibt sich Formel 5-3. Daraus ist zu erkennen, dass sich die Querschnittsfläche herauskürzt und somit keine Rolle spielt.

$$JPU = w \left[\frac{\text{m}}{\text{s}} \right] * 60 * \frac{273,15}{298,15} * \left(\frac{H[\text{mm}] * 9,80665}{\Delta p[\text{Pa}]} \right)^{0,6}$$

Formel 5-3

In der nachfolgenden Tabelle 18 sind die wichtigsten Ergebnisse aus den Simulationen aufgelistet und den experimentellen Werten gegenübergestellt. Dabei fällt auf, dass der Unterschied des spezifischen Druckverlustes zwischen [5] und den Simulationen erstaunlich gering ist. Selbst bei höheren Geschwindigkeiten weichen die Ergebnisse nur geringfügig voneinander ab. Auch das Gegenüberstellen der Permeabilität in Z-Richtung bestätigt die Ähnlichkeit beider Analysemethoden, da hier die relativen Differenzen gerade einmal ca. 4 % betragen. Zu Beginn steigen die Werte in beiden Fällen kontinuierlich an, sinken dann allerdings bei der höchsten Geschwindigkeit ($v = 1,6 \text{ m/s}$) wieder ab. Der Grund liegt in dem stetig steigenden Druckverlust, welcher in Formel 5-3 zur Berechnung der Permeabilität in JPU im Klammerausdruck unter dem Bruchstrich steht. Ab einem gewissen Zeitpunkt fällt der Klammerausdruck stärker, als die Geschwindigkeit steigt und die Werte sinken. Insgesamt sind große Übereinstimmungen in den Parametern erkennbar.

Tabelle 18: Ergebnisse aus der Strömungssimulation des Modells „Kugel“ und Vergleich mit den experimentellen Daten aus [5]

Ergebnisse aus den Simulationen										
	Kugel	[5]	Kugel	[5]	Kugel	[5]	Kugel	[5]	Kugel	[5]
Geschwindigkeit [m/s]	v = 0,25		v = 0,6		v = 1		v = 1,3		v = 1,6	
Reynoldszahl [-]	26,7	27,2	64,1	65,5	106,8	108,6	138,8	141,2	170,9	173,8
$\Delta p/H$ [Pa/m]	934	1.033	3.505	3.342	7.704	7.246	11.122	10.978	16.021	15.546
Permeabilität [JPU]	56,3	54,0	61,1	64,1	63,5	67,1	66,3	68,0	65,5	67,9

Neben der numerischen Berechnung bietet GeoDict die Möglichkeit den Geschwindigkeitsverlauf, als auch die Druckunterschiede innerhalb der Packung darzustellen. Abbildung 34 zeigt links eine dreidimensionale Illustration von „Kugel“, wobei die Z-Achse und somit die Strömungsrichtung von unten nach oben verläuft. Die schwarzen Punkte stellen die Partikel dar. Die mittlere Geschwindigkeit in diesem Szenario beträgt 1 m/s , erreicht aber während des Durchströmens sogar eine maximale Geschwindigkeit von $21,95 \text{ m/s}$. Anhand der Farbskala, welche in den Abbildungen jeweils rechts zu sehen ist, ist zu erkennen, dass dieses Maximum allerdings nur im Eintrittsbereich auftritt und der Fluss dann schnell an Tempo verliert. Auch innerhalb des Modells sind nur leichte Verwirbelungen zu erkennen, sodass die Farbgebung im Großen und Ganzen im dunklen Blaubereich bleibt. Rechts ist der Druckverlust über die Schüttung in einer zweidimensionalen Ansicht dargestellt. Der Schnitt durch den Quader erfolgt dabei entlang der ZY-Ebene, wobei die linke Illustration die vorderste Schnittebene darstellt und rechts daneben der Schnitt mittig der X-Länge erfolgt. Es zeichnet sich dasselbe Bild ab, welches schon bei dem Geschwindigkeitsverlauf zu sehen ist. Obwohl die Werte in einem Bereich zwischen $-130 \leq p \leq 250 \text{ Pa}$ liegen, sind kaum Druckunterschiede zu erkennen. Diese finden sich am ehesten lokal in der vordersten Schnittebene der ZY-

Fläche. Auch hier sind die hohen Werte am Eintritt erkennbar, während das mittlere Areal hauptsächlich eine türkise Farbe annimmt und somit die Werte laut der Farbskala um den Mittelwert schwanken. Im Austritt links sind sogar negative Drücke und somit Unterdruck sichtbar.

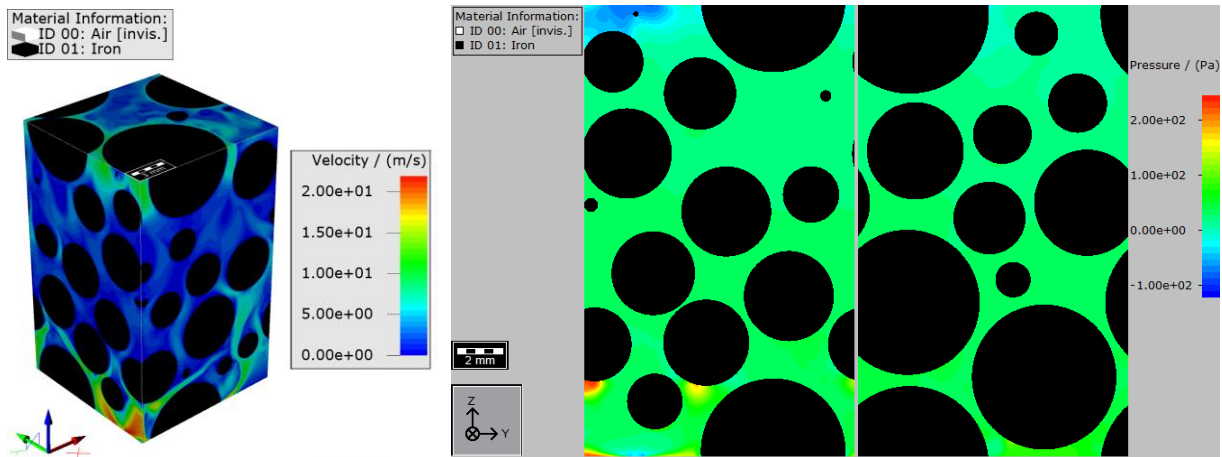


Abbildung 34: Links: Dreidimensionale Darstellung des Geschwindigkeitsverlaufs innerhalb des Modells „Kugel“; Rechts: Verlauf des Druckverlustes in der Packung in der zweidimensionalen Ansicht – zwei verschiedene Schnittebenen (Front und Mitte)

Die nachfolgende Abbildung 35 stellt die unterschiedlichen Druckverlustkurven im Vergleich dar, das heißt den spezifische Druckverlust in Abhängigkeit der Durchströmungsgeschwindigkeit. Verglichen werden darin die Ergebnisse aus der Simulation mit den experimentellen Daten aus [5], sowie den empirischen Druckverlustgleichungen, welche in Kapitel 3.2. beschrieben sind. Bei der Bezeichnung Ergun-Gleichung handelt es sich um Formel 3-21, deren Konstanten aus [16] entnommen sind. Die Werte für die Stoffparameter der Luft stammen aus der Tabelle 17 und der aus der Korngrößenverteilung berechnete Sauterdurchmesser ist in Tabelle 3 zu finden. Die sich daraus ergebende Druckverlustkurve ist im Diagramm in grüner Farbe dargestellt, weicht allerdings sehr stark von den Simulationsergebnissen ab, welche in rot eingefärbt sind. Bei gleichen Stoffwerten und Konstanten müsste der Sauterdurchmesser in der Ergun-Gleichung rund 4,2 mm betragen, um auf etwa derselben Höhe wie die numerischen und experimentellen Werte zu liegen. Formel 3-20 wiederum bezieht sich auf die spezifische Oberfläche der Korngrößenverteilung wie die Carman-Kozeny-Gleichung, weswegen diese Kurve im Diagramm mit „Ergun-Carman-Gleichung“ abgekürzt ist. Abgebildet sind zwei verschiedene spezifische Oberflächen, welche sich durch die unterschiedlichen Analysemethoden ergeben und ebenfalls der Tabelle 3 entnommen werden können. Die dunkelblaue Linie im Diagramm stellt dabei das Ergebnis bei Verwendung der minimalsten spezifischen Oberfläche dar ($a_{V,S} = 35,61 \text{ cm}^2/\text{cm}^3$), welche aus der photooptischen Analyse resultiert. Im Gegensatz dazu wird für die gelbe Linie der maximale Wert mit $a_{V,S} = 50,16 \text{ cm}^2/\text{cm}^3$ eingesetzt. Allerdings liegen auch hier beide Druckverlustverläufe oberhalb der Simulationsergebnisse, weichen aber nicht so stark ab wie das Ergebnis aus der Ergun-Gleichung. Eine Variation der spezifischen Oberfläche in „Ergun-Carman“ bei Erhaltung der Konstanten und Stoffwerte ergibt $27 \text{ cm}^2/\text{cm}^3$ für eine

Übereinstimmung mit der roten Kurve. Überraschend ist der fast perfekte Verlauf der experimentellen Ergebnisse mit den Simulationen. Dies lässt vermuten, dass es in der Probe bei dem Versuch entgegen der in [5] dargestellten Meinung zu keiner Verdichtung kam. Im Gegenteil scheinen sich im Inneren beim Einfüllen eher große Leerräume gebildet zu haben, welche die Druckverlustkurve im Gesamten schließlich so flach ausfallen ließen. Diese Vermutung wird durch die hohen Werte der Permeabilität bestätigt.

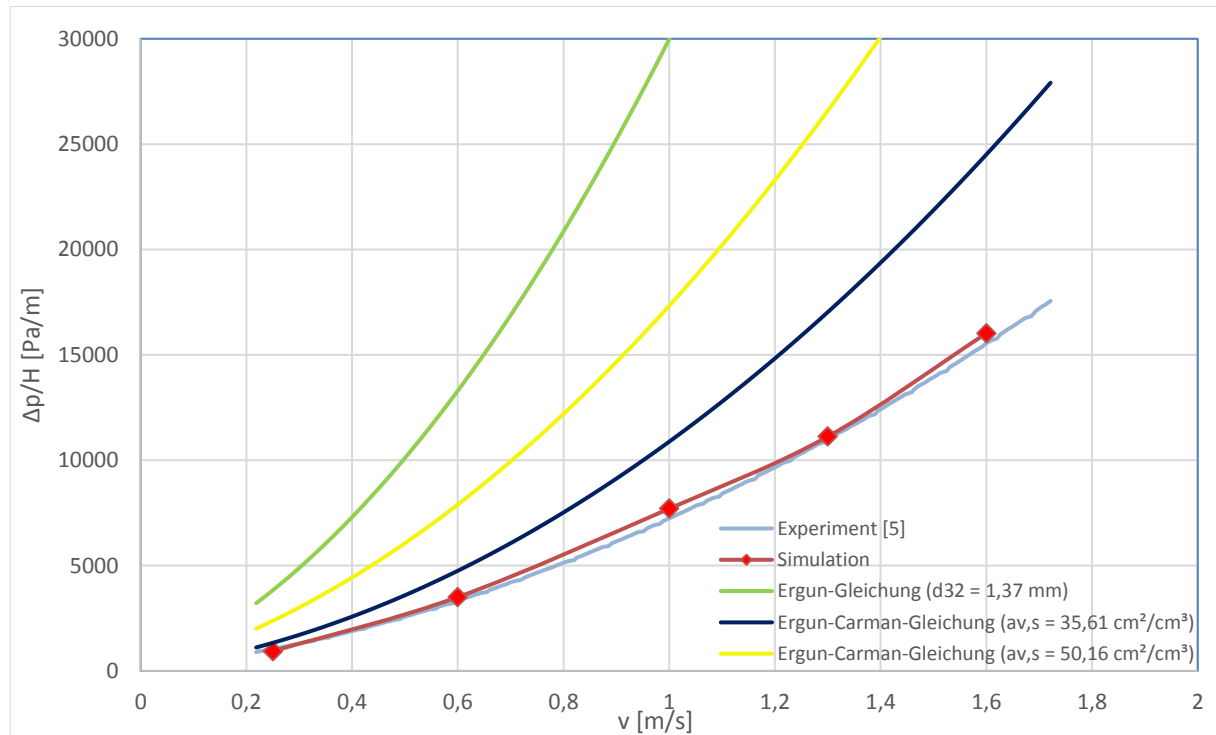


Abbildung 35: Druckverlust-Charakteristik von Modell „Kugel“ aus Abschnitt 2.1

5.2 Ergebnisse aus der Strömungssimulation der Packung „Polyeder-46“

Da dieses Modell auf derselben Korngrößenverteilung basiert, wie bereits bei der vorangegangenen Packung „Kugel“, können dieselben experimentellen Ergebnisse zum Vergleich herangezogen werden. Aus diesem Grund gelten die im Kapitel 4.1. am Anfang dargeführten Erläuterungen bezüglich der Durchführung der Versuche und deren Verwendung für vergleichende Darstellungen auch hierbei.

Obwohl dieses Modell dieselbe Porosität und Korngrößenverteilung wie „Kugel“ aufweist, zeigen sich deutliche Unterschiede bei dem Vergleich mit den experimentellen Ergebnissen. Die spezifischen Druckverluste weichen bereits bei dem ersten Simulationspunkt ($v = 0,25$ m/s) um fast 30 % voneinander ab. Die Differenz zwischen den numerischen und experimentellen Werten nimmt weiter zu bis zu einer Abweichung von etwa 50 %. Ähnlich dem Druckverlust zeigt auch die numerische Permeabilität deutliche Diskrepanzen zu [5], welche rund 30 % niedriger liegen. Kontinuierlich steigen allerdings wie bei dem Modell „Kugel“ auch

hier die Werte nicht an mit steigender Geschwindigkeit. Bereits ab dem zweiten Simulationsdurchgang sinkt die Permeabilität sogar wieder stetig.

Tabelle 19: Ergebnisse aus der Strömungssimulation des Modells „Polyeder-46“ aus Abschnitt 2.2 und Vergleich mit den experimentellen Daten aus [5]

Ergebnisse aus den Simulationen										
	Poly- eder- 46	[5]	Poly- eder- 46	[5]	Poly- eder- 46	[5]	Poly- eder- 46	[5]	Poly- eder- 46	[5]
Geschwindigkeit [m/s]	v = 0,25		v = 0,6		v = 1		v = 1,3		v = 1,6	
Reynoldszahl	26,7	27,2	64,1	65,5	106,8	108,6	138,3	141,2	170,9	173,8
$\Delta p/H$ [Pa/m]	1.442	1.033	5.641	3.342	13.421	7.246	21.346	10.978	30.596	15.546
Permeabilität [JPU]	43,4	54,0	46,0	64,1	45,5	67,1	44,8	68,0	44,4	67,9

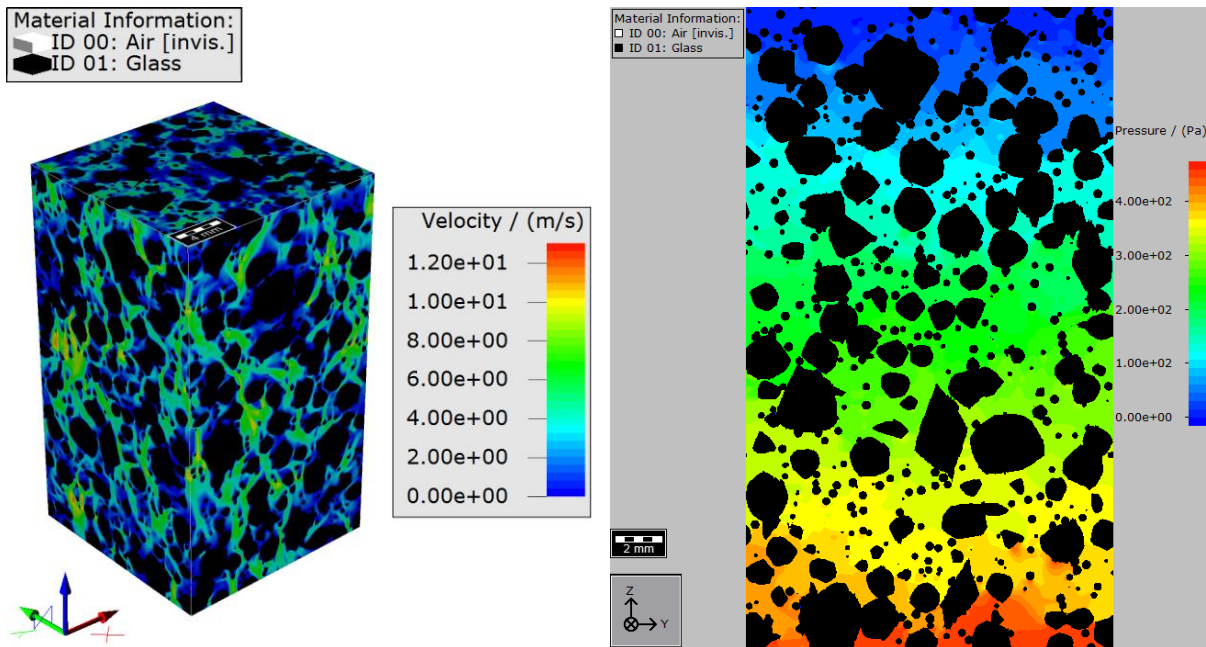


Abbildung 36: Links: Dreidimensionale Darstellung des Geschwindigkeitsverlaufs innerhalb des Modells „Polyeder-46“; Rechts: Verlauf des Druckverlustes in der Packung in der zweidimensionalen Ansicht (vorderste Schnittebene)

Eine dreidimensionale Darstellung des Geschwindigkeitsverlaufes innerhalb der Packung, welches in der nachfolgenden Abbildung 36 illustriert ist, zeigt ein ganz anderes Bild wie jenes des Modells „Kugel“ (Abbildung 34). Bei dem vorherigen Modell sind nur im Eintrittsbereich starke Differenzen erkennbar. Hierbei sind allerdings über die ganze Höhe hinweg Turbulenzen zwischen den Partikeln aufgrund der Durchströmung zu sehen. Den maximalen Wert erreicht die Luftgeschwindigkeit bei etwa 13,04 m/s, wobei hier ebenfalls die Durchschnittsgeschwindigkeit 1 m/s beträgt. Der Druckverlust ist rechts daneben abgebildet, wobei die zweidimensionale Darstellung die vorderste Seite der ZY-Ebene zeigt. Der Druck fällt innerhalb der Packung kontinuierlich, fast schichtweise, ab. Im Eintrittsbereich sind die

Maximalwerte erkennbar, welche bis zu 470 Pa annehmen können. Durch den Widerstand der Partikel sinkt der Druck allerdings auf rund -20 Pa ab. Ein direkter Vergleich des durchschnittlichen Druckverlaufs in der Packung über die Höhe ist in Abbildung 37 dargestellt. Aufgetragen ist der Druck in der Einheit Pa gegen die Z-Achse, welche hier in Voxel angegeben wird. Der Eintritt des Luftstromes erfolgt dabei bei Layer 1. Links ist „Kugel“ zu sehen, worin das Gefälle sehr sprunghaft und unregelmäßig verläuft. Lokal sind sogar positive Steigungen während des Verlaufs ersichtlich. Insbesondere im Eingangsbereich sind stärkere Differenzen, als mittig des Modells erkennbar. Rechts daneben ist der Druck innerhalb von „Polyeder-46“ dargestellt, welcher fast gleichmäßig über die gesamte Höhe abnimmt.

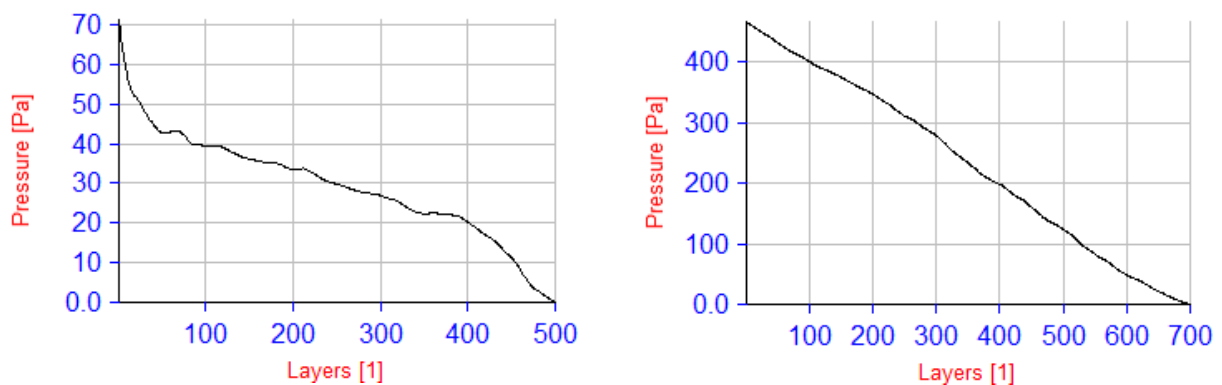


Abbildung 37: Verlauf des Drucks über die Höhe des Modells „Polyeder-46“ aus Abschnitt 2.2 (links) im Vergleich zu dem Verlauf in „Kugel“ aus Abschnitt 2.1

Die aus der Simulation resultierende Druckverlustcharakteristik ist in der Abbildung 38 dargestellt, wobei diese wieder mit den experimentellen Daten aus [5] und denselben empirischen Druckverlustgleichungen verglichen wird, wie schon in Kapitel 4.1. Stoffwerte und Konstanten werden in den Gleichungen übernommen und können für genauere Details auf S.67 nachgelesen werden. Wie schon zuvor erwähnt, übertreffen bei diesem Modell die Druckverlustwerte jene aus den praktischen Versuchen. Auch das Ergebnis aus der Ergun-Gleichung weicht von den numerischen Werten ab und würde sich erst bei einem Sauterdurchmesser von etwa 2,4 mm der roten Kurve angleichen. Überraschend ist die fast exakte Übereinstimmung der Simulationen mit der „Ergun-Garman-Gleichung“ mit einer eingesetzten spezifischen Oberfläche von rund $45,98 \text{ cm}^2/\text{cm}^3$. Dieser entstammt der grafischen Analyse, welche händisch durchgeführt wurde. Die anderen Werte für die spezifische Oberfläche aus Tabelle 3 werden aufgrund der hervorragenden Ähnlichkeit nicht in das Diagramm eingefügt.

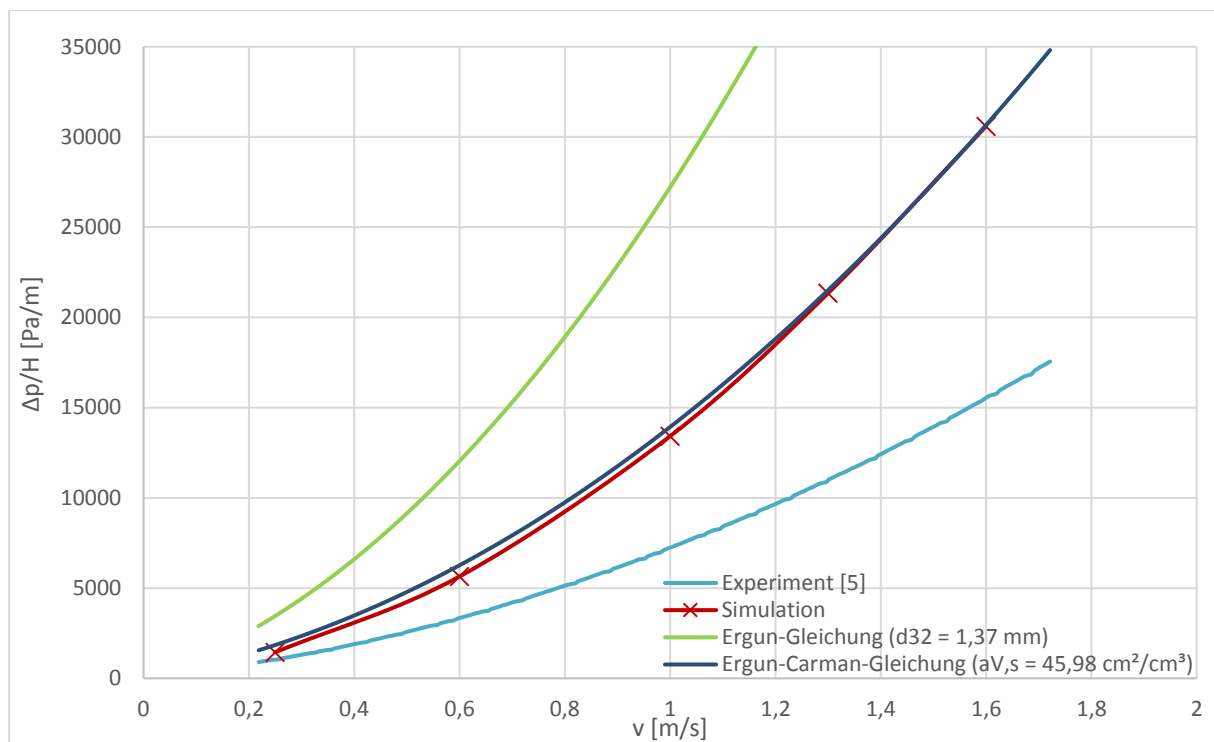


Abbildung 38: Druckverlust-Charakteristik von Modell „Polyeder“ aus Abschnitt 2.2

5.2.1 Variation der Packung „Polyeder-46“ durch Entfernen der kleinsten Kornformgruppe

Eine besondere Eigenschaft des „Polyeder-46“-Modells wird durch dessen stufenweise Generierung hervorgerufen, welche im Kapitel 2.2. detailliert beschrieben ist und die Körner in Kornformgruppen aufteilt. Diese besitzen unterschiedliche Material ID's, wodurch eine leichte, nachträgliche Entfernung einer Gruppe aus dem fertigen Modell möglich ist. Zur Untersuchung des Einflusses der kleinsten Partikel auf den Druckverlust wird sich diese Eigenschaft zunutze gemacht.

Das Aussondern der Kornformgruppe Klein, welche einen Korngrößenbereich zwischen 0,1 und 0,5 mm umfasst, erfolgt durch ein einfaches Umändern der Material ID 04 in 0 (= Luft). Dadurch erhöht sich die Porosität um etwa 6 % auf 52,98 %, wobei die restliche Verteilung des Feststoffanteils dieselbe bleibt. In Tabelle 20 sind der neue Luftanteil, sowie die Anteile der Kornformgruppen aufgelistet. Nachfolgend ist in Abbildung 39 eine vergleichende Darstellung der alten Packung mit allen Kornformgruppen auf der linken Seite und rechts daneben das neue Modell ohne der kleinsten Gruppe zu sehen. Als Kurzbezeichnung wird dieses Modell im nachfolgenden nur mehr mit „Ohne Klein“ abgekürzt. Das Schnittbild zeigt die YZ-Ebene und verläuft mittig der X-Achse. In dem direkten Vergleich fallen zum einen auf der rechten Seite die hohe Anzahl der kleinen Körner auf, obwohl diese nur 6,86 % des gesamten Raumes einnehmen. Des Weiteren sind nach dem Entfernen Löcher und Kerben in den restlichen Partikeln zu sehen.

Tabelle 20: Veränderte Porosität und Packungsverteilung durch die Entfernung der kleinsten Kornformgruppe

Kornformgruppen und Porosität		
Porosität	52,98	[%]
Groß	3,42	[%]
Mittel-groß	16,3	[%]
Mittel-klein	27,29	[%]

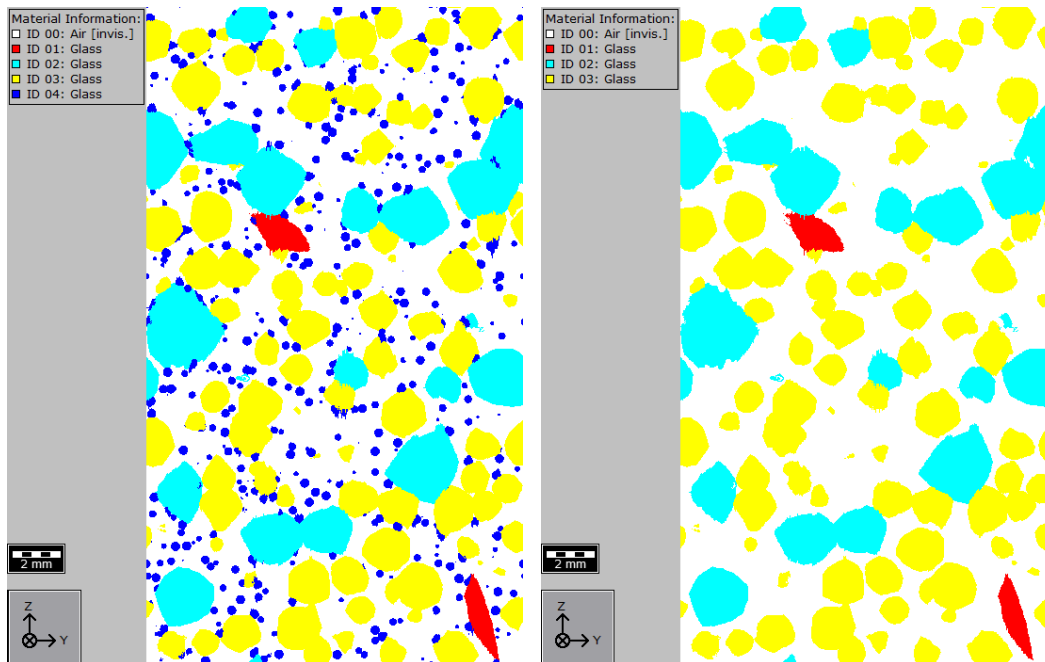


Abbildung 39: Links: 2D-Darstellung des Modells „Polyeder-46“; Rechts: 2D-Darstellung des Modells „Ohne Klein“ nach Entfernung der kleinsten Kornformgruppe

Die Ergebnisse aus den Simulationen, aufgelistet in Tabelle 21, zeigen den enormen Einfluss der Kornformgruppe Klein. Der spezifische Druckverlust ist signifikant gesunken und beträgt bei der ersten Simulationsdurchführung im Vergleich zu der ursprünglichen Packung ($dp/H = 1442 \text{ Pa}$) nur mehr rund 400 Pa . Dies ergibt eine Differenz von etwas mehr als 72% . Auch die experimentellen Werte weichen stark von der modifizierten Packung ab, denn deren Ergebnisse liegen im Durchschnitt etwa 50% darüber. Auch in der Permeabilität ergeben sich folglich starke Unterschiede, wenn die Werte miteinander verglichen werden. Interessant sind die unregelmäßig fallende und steigende Ergebnisse des Modells „Ohne Klein“, wodurch sich die relative Differenz zu den Daten aus [5] mit steigender Geschwindigkeit verringert.

Tabelle 21: Ergebnisse aus der Strömungssimulation des Modells „Ohne Klein“ und Vergleich mit den experimentellen Daten aus [5]

Ergebnisse aus den Simulationen										
	Ohne Klein	[5]	Ohne Klein	[5]	Ohne Klein	[5]	Ohne Klein	[5]	Ohne Klein	[5]
Geschwindigkeit [m/s]	v = 0,25		v = 0,6		v = 1		v = 1,3		v = 1,6	
Reynoldszahl	26,7	27,2	64,1	65,5	106,8	108,6	138,8	141,2	170,9	173,8
$\Delta p/H$ [Pa/m]	400	1.033	1.480	3.342	3.758	7.246	6.883	10.978	8.931	15.546
Perm [JPU]	93,7	54,0	102,6	64,1	97,7	67,1	88,4	68,0	93,0	67,9

Das nachfolgende Druckverlust-Diagramm (Abbildung 40) zeigt abermals deutlich den enormen Unterschied durch die Entfernung der kleinsten Kornformgruppe. Dargestellt sind dieses Mal allerdings nur die charakteristische Kurve des Modells „Polyeder-46“, „Ohne Klein“ und die experimentellen Versuchsdaten. Eine Modifikation der Eingabevariablen für den Sauterdurchmesser und der spezifischen Oberfläche in der „Ergun-Gleichung“ und „Ergun-Carman-Gleichung“ ergeben Sollwerte von $d_{32} \approx 7 \text{ mm}$ und $a_{V,S} \approx 17 \text{ cm}^2/\text{cm}^3$ für eine passende Übereinstimmung.

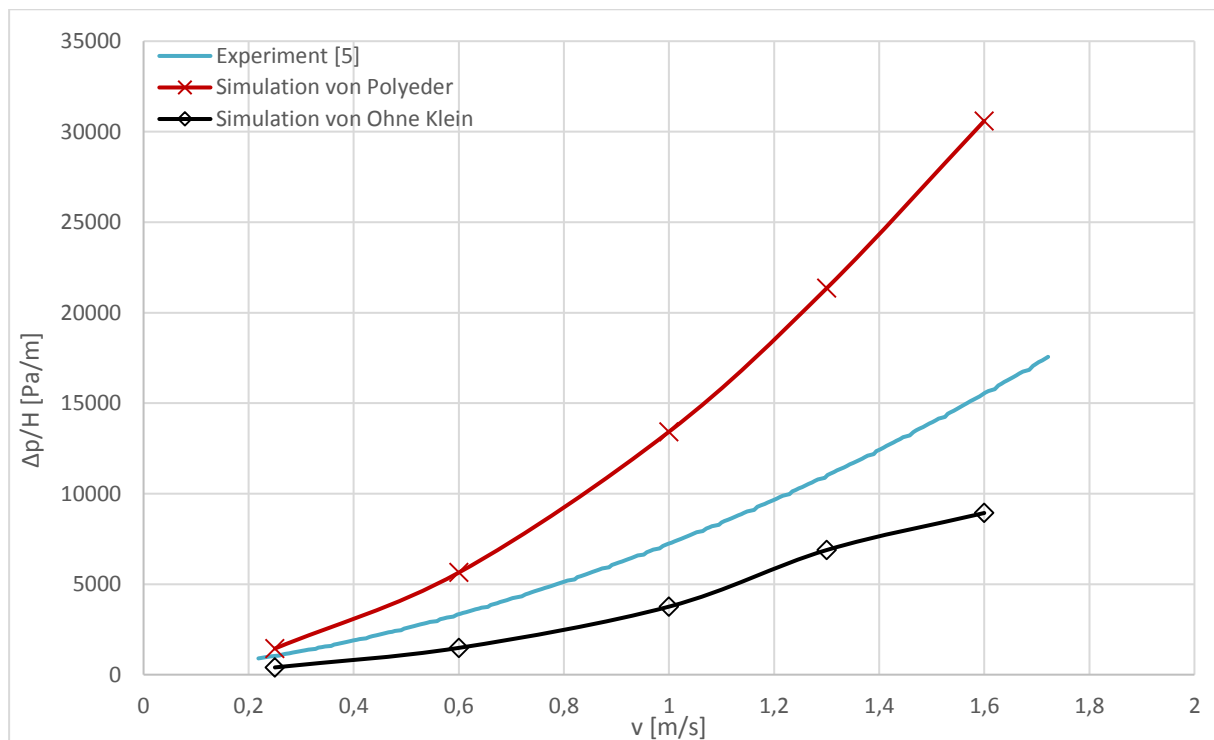


Abbildung 40: Druckverlust-Charakteristik von Modell „Ohne Klein“

5.3 Ergebnisse der Strömungssimulation der Packung „Polyeder-36“

Da in diesem Fall das Modell „Polyeder-36“ nicht mehr auf derselben Korngrößenverteilung, wie die vorherigen Modelle basiert, können auch nicht dieselben Versuchsdaten zum Vergleich herangezogen werden. Stattdessen stehen aber Versuchsdaten aus einer Permeabilitätsmessung zur Verfügung, welche im Rahmen der Arbeit von [25] durchgeführt wurden. Die untersuchte Feststoffprobe wurde direkt dem Mischgutstrom entnommen, welcher auf das Sinterband herunterfiel und hinsichtlich der enthaltenen Feuchte analysiert, bevor es dem Permeabilitäts-Messgerät zugeführt wurde. Die Funktionsweise verwendeten Apparatur ist dabei gleich jener, welche in Kapitel 4.1 anfangs beschrieben wird. Allerdings wird der Sollwert des Luftvolumenstroms während der Versuchsdurchführung konstant gehalten, welcher rund $0,46 \text{ Nm}^3/\text{min}$ beträgt. Umgerechnet entspricht dieser Wert wiederum einer Luftgeschwindigkeit von etwa 1 m/s . Zur Verfügung steht daher keine Druckverlustkurve, sondern nur ein Schwankungsbereich des Druckverlustes für einen Geschwindigkeitswert. Auch die Porosität selbst kann nicht direkt miteinander verglichen werden, da diese nicht gemessen wurde. Zur Verfügung stehen allerdings Messergebnisse für die Schüttdichte bei jedem Versuch. Daraus lässt sich über die Rohdichte nach Formel 2-17 ein Wert für die Porosität angeben. Die Rohdichte wird mit $\rho_s = 3,29 \text{ g/cm}^3$ aus [5] entnommen. Gemittelt ergibt sich somit eine Porosität von etwa $41,4 \%$, wodurch diese experimentelle Schüttung in ihrer Kompaktheit zwischen der Porosität von „Polyeder-36“ und „Polyeder-46“ steht.

Für die Strömungssimulationen selbst werden wieder dieselben Eingabewert für das Modul FlowDict der Software GeoDict verwendet, welche in Tabelle 17 zusammengefasst sind. Da sich aufgrund der Verwendung einer anderen Korngrößenverteilung im Zuge dessen auch die charakteristische Länge ändert, kann nicht mehr der Wert wie vorher herangezogen werden. Stattdessen wird der harmonischer Durchmesser anhand der Daten aus Tabelle 13 mittels Formel 3-4 neu berechnet, wodurch sich ein Wert von $2,78 \text{ mm}$ (zum Vergleich: vorher $d_{\text{harm}} = 1,67 \text{ mm}$) ergibt. In Tabelle 22 sind die wichtigsten Ergebnisse aus den Strömungssimulationen aufgelistet, wobei wie oben bereits erwähnt nur die numerischen Werte für $v = 1 \text{ m/s}$ mit einer praktischen Versuchsdurchführung verglichen werden können. Diese werden als Mittelwert dargestellt, wobei die Minimal- und Maximalwerte sowohl für den spezifischen Druckverlust, als auch der Permeabilität nur rund 7% davon entfernt sind. Der Schwankungsbereich ist somit sehr eng. Beim Vergleich der numerischen und experimentellen Ergebnisse zeigt sich allerdings eine starke Abweichung. Der spezifische Druckverlust ist fast 4-mal so hoch und auch die Permeabilität zeigt eine relative Differenz von fast 60% . Dies scheint allerdings in Anbetracht der unterschiedlichen Porositäten nicht verwunderlich.

Tabelle 22: Ergebnisse aus der Strömungssimulation des Modells „Polyeder-36“ aus Abschnitt 2.3 und Vergleich mit den experimentellen Daten aus [25]

Ergebnisse aus den Simulationen						
	Polyeder 36	Polyeder 36	Polyeder 36	[25]	Polyeder 36	Polyeder 36
Geschwindigkeit [m/s]	$v = 0,25$	$v = 0,6$	$v = 1$		$v = 1,3$	$v = 1,6$
Reynoldszahl	44,4	106,7	177,8		231,1	284,4
$\Delta p/H$ [Pa/m]	8.807	35.022	85.149	21.508	137.101	200.341
Permeabilität [JPU]	14,7	15,4	15,0	36,5	14,7	14,4

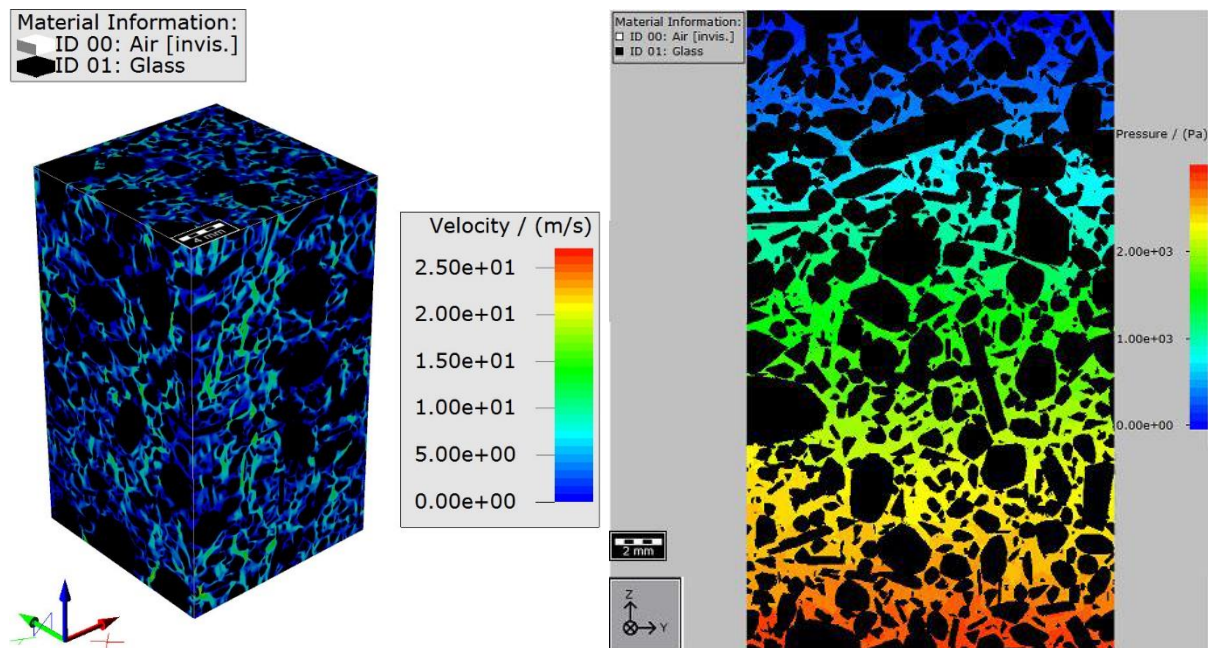


Abbildung 41: Links: Dreidimensionale Darstellung des Geschwindigkeitsverlaufs innerhalb des Modells „Polyeder-36“; Rechts: Verlauf des Druckverlustes in der Packung in der zweidimensionalen Ansicht (vorderste Schnittebene)

Nachfolgend sind wieder Darstellungen des Druck- (rechts) bzw. des Geschwindigkeitsverlaufes (links) innerhalb der Packung in Abbildung 41 illustriert. Dafür wurde abermals eine Durchschnittsgeschwindigkeit von 1 m/s ausgewählt und die Partikel schwarz eingefärbt. Durch den höheren Widerstand während des Durchströmens ergeben sich stärkere Unterschiede, als beim Modell „Polyeder-46“. Das maximale Flusstempo ist hier um etwa das doppelte auf 27,1 m/s gestiegen, wobei dies nur sehr punktuell auftritt. Bei näherer Betrachtung der linken Illustration in Abbildung 41 sind kaum Farbunterschiede zu erkennen. Die Packung ist überwiegend dunkelblau, was auf einen niedrigen Geschwindigkeitswert hinweist. Selbst die auftretenden, helleren Flecken in grün sind abgedunkelt und stechen kaum heraus. Rechts zeigt der Druckverlauf ein ähnliches Bild, wie jenes des Modells „Polyeder-46“. Die zweidimensionale Darstellung rechts, welche die vorderste Schnittebene der YZ-Ebene zeigt, lässt sich wieder in horizontale Abschnitte unterteilen, in denen die Druckwerte von

unten nach oben stetig abnehmen. Dabei tritt ein Unterdruck bis zu etwa 70 Pa auf. Maximal wird ein Überdruck von rund 3000 Pa erreicht.

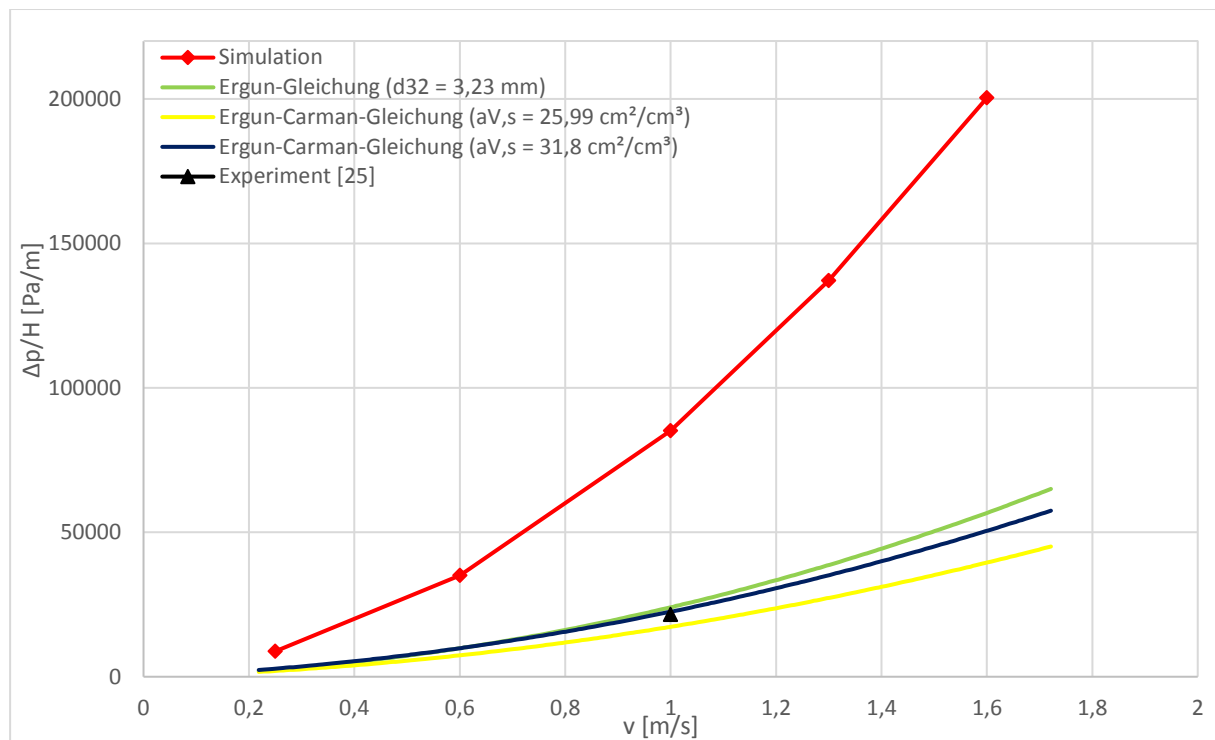


Abbildung 42: Druckverlust-Charakteristik von Modell „Polyeder-36“ aus Abschnitt 2.3

In Abbildung 42 ist wieder der Druckverlustverlauf aus den numerischen Simulationen vergleichend zu den empirischen Gleichungen und den experimentellen Daten aus [25] dargestellt. Letzteres ist als schwarzes Dreieck gekennzeichnet und überlappt sich mit der „Ergun-Carman-Gleichung“ mit einer spezifischen Oberfläche von $31,8 \text{ cm}^2/\text{cm}^3$. Dieser Wert entstammt der grafischen Analyse anhand des RRSB-Diagramms, welches in Anhang II zu finden ist. Die anderen beiden Werte für die spezifische Oberfläche ($a_{V,s} = 25,99 \text{ cm}^2/\text{cm}^3$) und Sauterdurchmesser ($d_{32} = 3,23 \text{ mm}$) wurden über Formel 2-4, Formel 2-5, Formel 2-7 und Formel 2-13 berechnet. Der verwendete Formfaktor beträgt 1,4 und kann für nähere Details auf S. 31 nachgelesen werden. Die bisherigen Bezugswerte für die Stoffparameter und Konstanten werden auch hierbei angewendet. Keine der empirischen Gleichungen nähert sich allerdings an die Simulationen heran. Alle weichen mit mehr als 70 % deutlich davon ab, sind allerdings zueinander sehr ähnlich. In den anderen Druckverlust-Charakteristiken aus den vorherigen Unterkapiteln aus Kapitel 5 waren starke Unterschiede durch die Verwendung des Sauterdurchmessers oder der spezifischen Oberfläche zu sehen. Um die empirischen Gleichungen anzunähern wären ein Sauterdurchmesser von etwa 1,2 mm oder eine spezifische Oberfläche von rund $80 \text{ cm}^2/\text{cm}^3$ notwendig.

5.4 Druckverlustcharakteristik der CT-basierten Sinterstruktur

Nach ihrer Modifikation werden alle Sinterstrukturen, welche in Abbildung 28 dargestellt sind, zusätzlich numerisch durchströmt und aus den Ergebnissen, wie bei den anderen Modellen, eine Druckverlustkurve erstellt. Die originale Struktur mit einer Porosität von 23,6 % kann allerdings nicht durchströmt werden, da der Quader in alle Richtungen keine durchgängigen Wege besitzt. Erst die Erhöhung der Porosität ermöglicht die Simulationen. Die Stoffparameter der Luft, sowie Randbedingungen und Eigenschaften des Solvers entsprechen dabei ebenfalls größtenteils den vorherigen Simulationendurchführungen und können in der Tabelle 17 nachgelesen werden. Unterschiede bestehen in nur in der Länge des „Inflow“ und „Outflow“ des „Add Implicit Region“, für welche 27 und 28 Voxel eingesetzt werden. Damit ergibt sich gesamt wieder ein Wert von 256, welcher wiederum der achten Potenz von zwei entspricht. Des Weiteren wird, ähnlich zum Modell „Polyeder-36“, aufgrund mangelnder Informationen als charakteristische Länge für die Berechnung der Reynoldszahl die Option „Numerical Length“ gewählt.

Tabelle 23: Ergebnisse von spezifischen Druckverlust, Reynoldszahl und Permeabilität aus Simulationen für Sinterstrukturen mit unterschiedlicher Porosität

Ergebnisse aus den Simulationen						
		v = 0,25	v = 0,6	v = 1	v = 1,3	v = 1,6
$\epsilon = 28,18 \%$	Reynoldszahl	0,04	0,08	0,11	0,12	0,14
	$\Delta p/H$ [Pa/m]	588.641	2.522.204	6.042.407	11.703.047	16.766.033
	Perm [JPU]	1,18	1,18	1,16	1,02	1,01
$\epsilon = 33,32 \%$	Reynoldszahl	0,11	0,19	0,27	0,32	0,37
	$\Delta p/H$ [Pa/m]	106.423	453.299	1.064.920	1.636.246	2.317.205
	Perm [JPU]	3,29	3,31	3,30	3,32	3,31
$\epsilon = 40,06 \%$	Reynoldszahl	0,23	0,42	0,58	0,68	0,77
	$\Delta p/H$ [Pa/m]	22.369	94.005	227.661	361.181	521.440
	Perm [JPU]	8,38	8,50	8,33	8,21	8,11
$\epsilon = 45,27 \%$	Reynoldszahl	0,34	0,67	0,92	1,08	1,22
	$\Delta p/H$ [Pa/m]	10.098	36.926	89.459	142.626	207.217
	Perm [JPU]	13,50	14,89	14,59	14,34	14,10

In Tabelle 23 sind die Ergebnisse der modifizierten Sinterstrukturen aus den Simulationen aufgelistet. Wie zu erwarten, sinkt der spezifische Druckverlust bzw. erhöht sich die Permeabilität mit steigender Porosität. Interessant ist dabei jedoch die relative Differenz zwischen den Packungen untereinander, zum Beispiel, die Ergebnisse aus $\epsilon = 28,18\%$ und $33,32 \%$ weichen konstant $84 \pm 2\%$ voneinander ab. Ein Vergleich der nächsten zwei Packungen zeigt eine stetige Abweichung von $78 \pm 1\%$. In Abbildung 43 sind nochmals die Druckverlustkurven aller Sinterstrukturen illustriert und zeigen die deutlichen Unterschiede trotz annähernd gleicher Zunahme des Porenanteils. Eine genauere Betrachtung der

Permeabilität lässt, ähnlich wie bei den Ergebnissen aus den vorherigen Unterkapiteln, keine Vorhersagen oder einheitlichen Trend erkennen. Für $\varepsilon = 28,18\%$ sinken die Werte teilweise, während für die nächsthöhere Porosität die Permeabilität abwechselnd steigt und fällt.

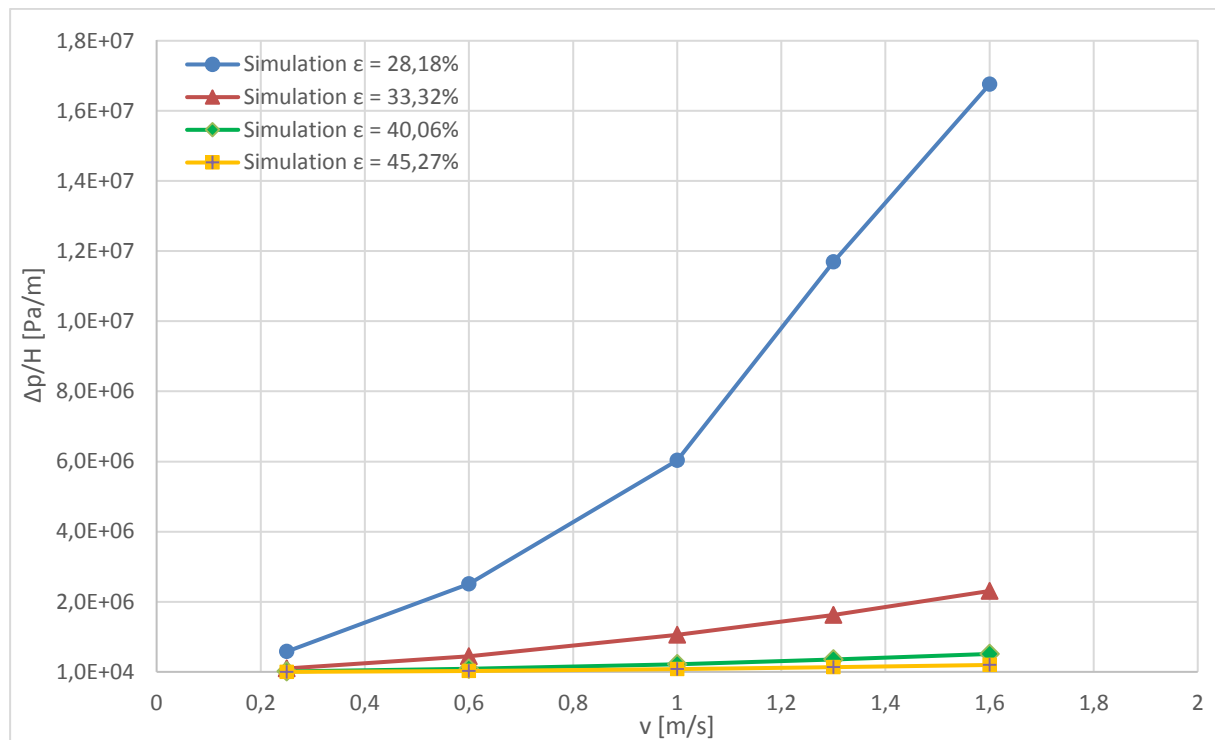


Abbildung 43: Vergleich der Druckverlustkurven aller CT-basierten, modifizierten Sinterstrukturen

Der modifizierte CT-Scan mit einer Porosität von 45,27 % lässt sich aufgrund seinem ähnlichen Porenanteil mit den Ergebnissen des Modells „Polyeder-46“ und den experimentellen Daten aus [5] gegenüberstellen. Verglichen werden in Abbildung 44 die unterschiedlichen Druckverlust-Charakteristika, wobei darin enorme Unterschiede zu erkennen sind. Am Ende von Kapitel 4.2 sind bereits die Differenzen von „Polyeder“ und den experimentellen Daten diskutiert worden. Die Sinterstruktur zeigt trotz nur etwas geringerer Porosität eine sehr starke Abweichung der Druckverluste. Ein Vergleich der Relationen zu der grünen Linie („Polyeder“) ergibt eine konstante Diskrepanz von etwa 85 % und zur blauen Linie ([5]) etwa 91 %. Der Grund dafür liegt wahrscheinlich an den engen Verzweigungen innerhalb des CT-Scans, welche sich durch die Manipulation der Porosität ergeben. Die Partikel in dem virtuellen Modell sind viel gleichmäßiger verteilt. Untersuchungen des Druckverlustes in der Packung haben bereits gezeigt, dass dieser im Querschnitt der XY-Ebene kaum schwankt (siehe Abbildung 36).

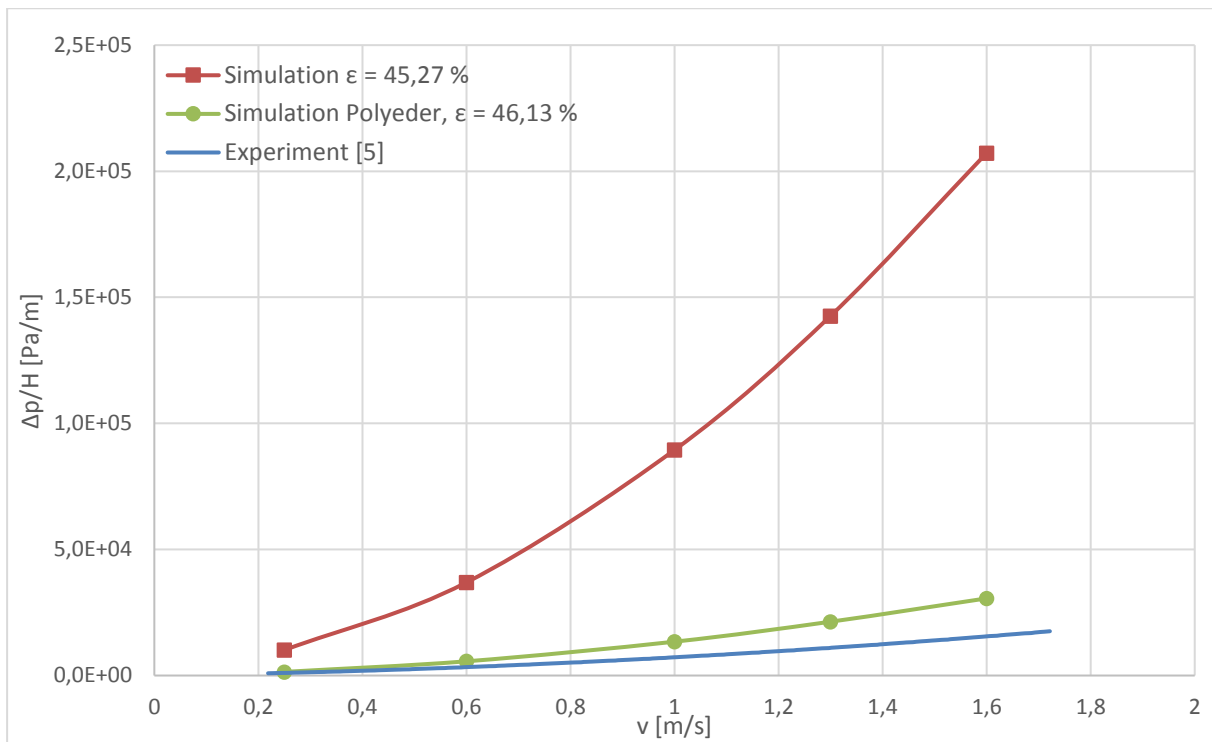


Abbildung 44: Vergleich der Druckverlust-Charakteristik von dem Modell „Polyeder-46“, der CT-basierten Sinterstruktur mit $\epsilon = 45,27 \%$ und dem Datensatz aus [5]

6 Zusammenfassung

Dass der Feuchteanteil in der Rohmischung Einfluss auf das Sintern hat, ist seit langem bekannt. Unbeachtet schien allerdings bisher der Wassergehalt für jene Proben, welche für Korngrößenanalysen herangezogen wurden. Durch das Vergleichen vieler verschiedener Proben konnte festgestellt werden, dass eine zu trockene Mischung den Feinanteil im Siebergebnis deutlich erhöht. Eine bewusste Trocknung, aber auch falsche oder zu lange Lagerung führt zu einer verfälschten Korngrößenverteilung. Da dies die Grundlage ist für die Ermittlung durchströmungsbezogener, weiterer Parameter, ist hierbei auf eine korrekte Durchführung zu achten. Anhand der Analyse können, z.B. wichtige Strukturparameter wie der Sauterdurchmesser oder die spezifische Oberfläche berechnet werden. Kleine Unterschiede in diesen Werten haben große Auswirkung beim Verwenden in empirischen Durchströmungsgleichungen. Bei den empirischen Druckverlustgleichungen selbst konnte keine Präferenz festgestellt werden. In den meisten Fällen wichen diese stark von den numerischen Simulationen ab. Überraschend war bei den numerischen Ergebnissen allerdings der große Einfluss der Kleinstfraktion mit einer Korngröße < 1 mm auf den Druckverlust. Obwohl diese nur einen kleinen Bruchteil (6,86 % vom gesamten Raum) ausmachen, zeigen die Ergebnisse nach dem Entfernen dieser Korngruppe eine Verringerung des Druckverlustes von über 70 % gegenüber dem Ausgangswert. Trotz ihrer geringen Größe ist der Feinkornanteil relevant für den Strömungswiderstand. Auch die Kornform spielt eine entscheidende Rolle beim Durchströmen der Packung. Bei einer perfekten Kugel ist der Widerstand deutlich geringer als bei einer kantigen, polyedrischen Form. Eine genauere Untersuchung der einzelnen Partikel in einer Schüttung ergab allerdings kaum runde Körner, weswegen eine Simulationen basierend auf rein sphärischen Formen als stark vereinfachend anzusehen ist.

Mit den in dieser Arbeit dargestellten Varianten zur Generierung einer virtuellen Schüttung aus dem Sinter-Rohmaterial kann eine genauere Beschreibung der Einflussfaktoren bei der Packungs-Durchströmung im Vergleich zu deren Quantifizierung alleinig auf Basis einfacher Durchströmungs- bzw. Druckverlustgleichungen erreicht werden. Obwohl diese Arbeit auf Grünmix-Schüttungen beim Sintern von Eisenerzen konzentriert, wäre es möglich die vorgestellten Methoden bei anderen Packungen aus geschütteten, komplex geformten Partikeln anzuwenden. Die hier dargestellten Ansätze bieten somit viel Potential für weitere Forschungsprojekte nicht nur im Bereich des Sinterns.

7 Verzeichnisse

7.1 Literatur

1. Bhattacharyya A., Höftberger A., Schenk J., Weiß C., Schaffer C., Schuster E., Modelling the dependence of sinter bed permeability on sinter porosity and structure (2018)
2. Cappel F., Wendeborn H., Sintern von Eisenerzen, Bd 19. Verl. Stahleisen, Düsseldorf (1973)
3. Drehmatrix. <https://de.wikipedia.org/w/index.php?oldid=179932487>. Zugegriffen: 12. Februar 2019
4. Ferziger J. H., Peric M., Numerische Strömungsmechanik. Springer-Verlag, Berlin Heidelberg (2008)
5. Griesser A. S., Einfluss der Mischgutfeuchte, der Korngrößenverteilung und weiteren agglomerationsrelevanten Korneigenschaften von Sintereinsatzmischungen auf den Druckverlust über die Schüttungen und die Sinterleistung. Masterarbeit, Montanuniversität Leoben (2016)
6. Haver&Boecker, Photooptische Partikelanalyse. Materialqualität genau erfassen. https://www.haver-partikelanalyse.com/fileadmin/02-b_Haver_Partikelanalyse/PA-Dokumente/P-66-D-scrgr_281114.pdf. Zugegriffen: 18. September 2018
7. Laurien E., Oertel H., Numerische Strömungsmechanik. Grundgleichungen und Modelle - Lösungsmethoden - Qualität und Genauigkeit ; mit über 530 Wiederholungs- und Verständnisfragen, 5. Aufl. Springer Vieweg (2013)
8. Lehner M., Skriptum Mechanische Verfahrenstechnik, Montanuniversität Leoben (2014)
9. Math2Market, EditGeo 2015. Reference (2015)
10. Math2Market, FlowDict. Release 2017 (2017)
11. Math2Market, GadGeo. Release 2017 (2017)
12. Math2Market, GrainGeo. Release 2017 (2017)
13. Math2Market, Material Database. Release 2017 (2017)
14. Math2Market, PoroDict. Release 2017 (2017)
15. Math2Market, ProcessGeo. Release 2017 (2017)
16. Menapace H. M., Untersuchungen zum Agglomerations- und Rollierverhalten des Intensivmischers der voestalpine Stahl Donawitz. Diplomarbeit, Montanuniversität Leoben (2004)
17. Mory M., Fluid Mechanics for Chemical Engineering. Wiley and sons, Hoboken, NJ USA (2013)
18. Müller W., Mechanische Verfahrenstechnik und ihre Gesetzmäßigkeiten, 2. Aufl. Studium. De Gruyter Oldenbourg, München (2014)
19. Ohser J., Mücklich F., Statistical analysis of microstructures in materials science. Wiley and sons (2000)
20. Rhodes M., Introduction to particle technology, 2. Aufl. Wiley and sons (2008)
21. Schenk J., Skriptum Eisen- und Stahlmetallurgie für IU und RT, Montanuniversität Leoben (2017)
22. Stieß M., Partikeltechnologie, 3. Aufl. Springer-Verlag (2009)
23. Svarovsky L. (Hrsg), Solid-liquid separation, 4. Aufl. Butterworth-Heinemann, Oxford (2000)
24. VDI-Wärmeatlas, 11. Aufl. Springer Reference. Springer Vieweg, Berlin (2013)
25. Voller N., Lindthaler M., Permeabilität und Festbettdurchströmung von mineralischen Schüttungen. Bakkalaureatsarbeit, Montanuniversität Leoben (2018)
26. Weiß C., Skriptum Stoffbilanzen und Stoffeigenschaften, Montanuniversität Leoben (2013)

7.2 Abkürzungsverzeichnis

A_S	Feststoffoberfläche
$a_{V,s}$	spezifische Oberfläche
$a_{V_{ges,s}}$	gesamte, benetzte Oberfläche
c_s	Schallgeschwindigkeit
d_{32}	Sauterdurchmesser
D_e	äquivalenter Durchmesser
d_{harm}	harmonischer Durchmesser
d_i	Durchmesser
d_{mi}	mittlere Korngröße einer Kornklasse
dp'	Lageparameter
d_s	oberflächenbezogener Durchmesser
d_V	volumenbezogener Durchmesser
f	Heywoodfaktor
f_x, f_y, f_z	Quellterme
H	Schütthöhe
H_e	äquivalente Höhe
JPU	Japanese Permeability Unit
k	Korrekturfaktor
K_1, K_2, K_3	Konstanten
K_{Bl}	gerätespezifische Konstante
K_S	dimensionslose Oberflächenkennzahl
Ma	Machzahl
n	Gleichmäßigkeitsparameter
\hat{n}	Einheitsvektor in beliebige Richtung
$N_X \times N_Y \times N_Z$	Voxelanzahl in X-, Y- und Z-Richtung
Δp	Druckverlust
Q	relative Durchgangssumme
ΔQ	relativen Durchgang
R	relativer Rückstand
$R_{\hat{n}}(\alpha)\vec{x}$	Rotationsmatrix von dem Einheitsvektor \hat{n} um den Winkel α mal dem Ausgangsvektor
s	Seitenlänge eines Voxels
Δt	Zeitdifferenz
T	Temperatur
T_N	Normtemperatur = 273,15°C
\dot{V}	Volumenstrom
V_H	Hohlvolumen
V_S	Feststoffvolumen
w	Geschwindigkeit
\vec{w}	Geschwindigkeit als Vektor
w_e	tatsächliche, effektive Geschwindigkeit

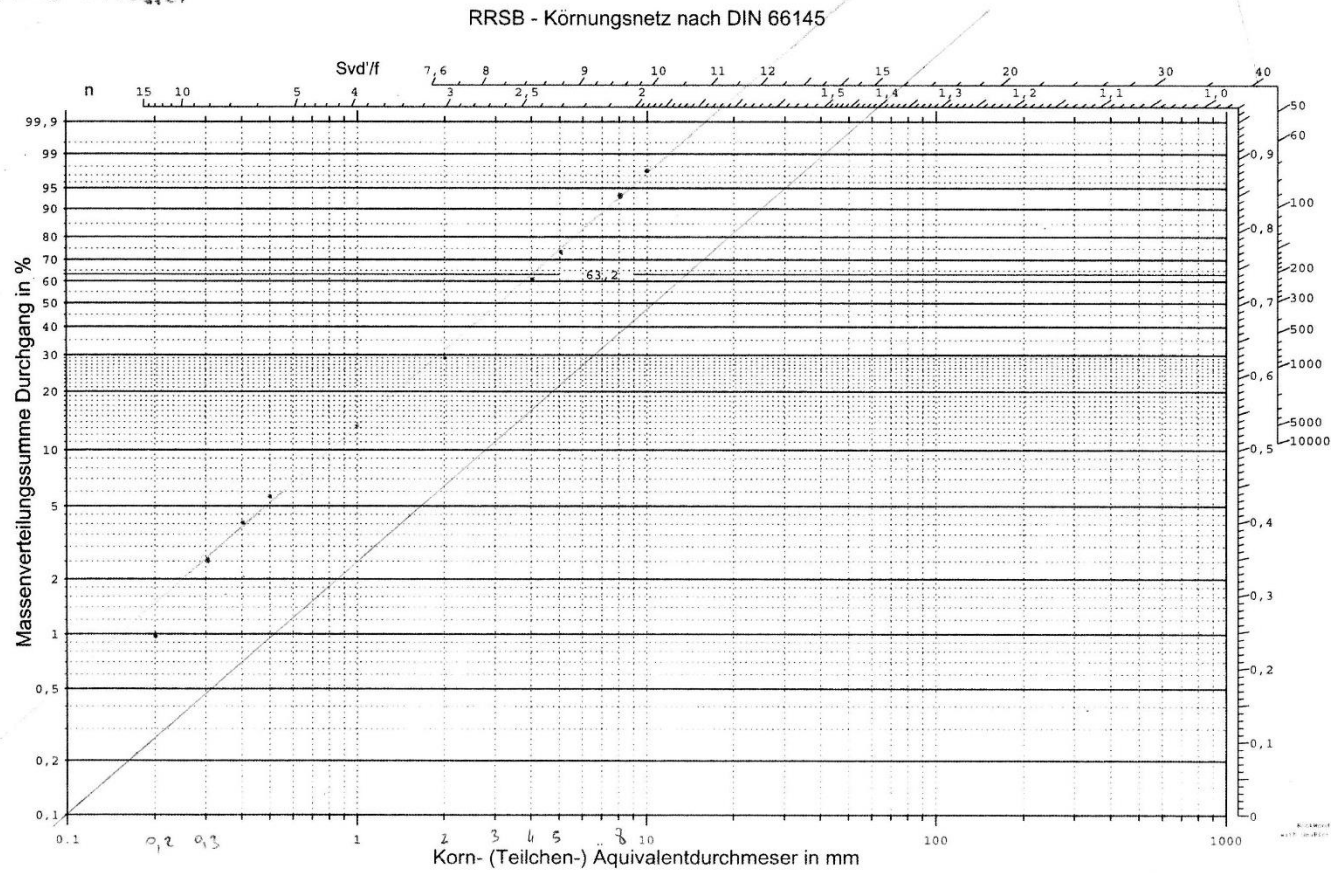
x	Korngröße
\vec{x}	Vektor eines Feststoffvoxels
z. B.	Zum Beispiel
$^{\circ}\text{C}$	Grad
α	beliebige Winkel
$\epsilon_{(\text{ges})}$	Porosität
ϵ_a	äußere Porosität
ϵ_i	innere Porosität
η_g	dynamische Viskosität
ν	kinematische Viskosität
ρ_{fl}	Dichte einer Flüssigkeit
ρ_g	Gasdichte
ρ_s	Feststoffdichte bzw. Rohdichte
$\rho_{\text{schütt}}$	Schüttdichte
φ	Formfaktor
Ψ	Sphärizität
∂	Partielle Ableitung

Anhang

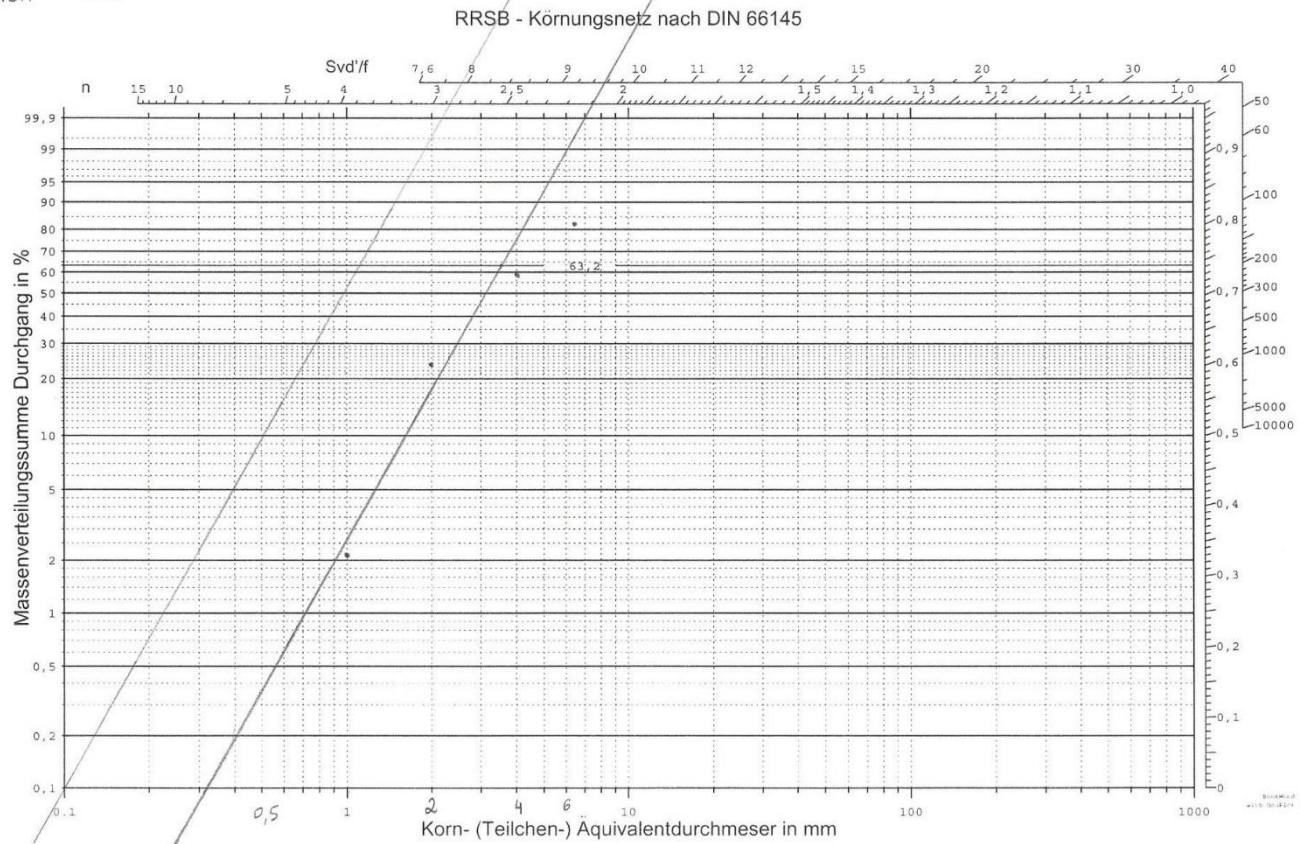
Probe 2 misch + roll

4. 1. 2016 in VOEST Linz

Von Anna Griesser



Grünbetsinter von ~~VEST~~ Donawitz
 Entnommen: 13.9.2014
 Analysiert: 14.9.2018 per Handrieblung
 von Voller & Lindthaler



	Beschreibung	Formfaktor
	Kugel	1,0
	Tropfen, Blase, rundes Korn	1,0–1,2
	eckiges Korn (z.B. Sand)	1,3–1,5
	nadelförmig	1,5–2,2
	plättchenförmig	2,5–4,0
	stark zerklüftete Oberfläche (z.B. Ruß)	100–10000

Tab. 2.3: Formfaktoren für einige Partikelformen

Python-Makro zum Einschleichen von Partikeln

```
Header = {
    'Release'      : '2017',
    'Revision'     : '19070',
    'BuildDate'    : '21 Jul 2017',
    'CreationDate' : '14 Sep 2017',
    'CreationTime' : '11:23:13',
    'Creator'      : 'vtiu',
    'Platform'     : '64 bit Windows',
}

Description = ""
Macro file for GeoDict 2017
recorded at 11:23:13 on Do Sep 14 2017
by Christian WEISS of Chair of Process Technology & Industrial Environmental Protection of Montanuniversität Leoben
""

Variables = {
    'NumberOfVariables' : 1,
    'Variable1' : {
        'Name'      : 'Project_Folder',
        'Label'     : 'Put in the name of the project folder',
        'Type'      : 'string',
        'Unit'      : "",
        'ToolTip'   : "",
        'BuiltinDefault' : 'New Project',
        'Check'     : ""
    }
}

#Explanations of variables syntax:
#Name:      mandatory, name of the variable by that it can be addressed in the code, must not contain white spaces!
#Label:     optional, appears as text in the input Dialog, if not given name is used
#Type:      mandatory, known types are bool, double, uint, int, string, filestring
#Unit:      optional, appears only in GUI (not used to rescale any input parameters automatically)
#           for type filestring, Unit contains the file suffix
#ToolTip:   optional, appears in GUI (must be in one line)
#BuiltinDefault: optional, default value which is used in macro (if not given, defaults to 0 or empty string)
#Check:     optional, known checks are positive, negative, min, max (checks are divided by semicolon)

import glob
import random
import os
import datetime

# -----Input-Box erstellen: Hier können die Korngrößenverteilung und Porosität eingegeben werden-----
-----
dlg1 = gd.makeDialog('Start', "Startparameter")
dlg1.addFloatInput("Porositaet", "Porositaet der Schuettung in %", min=0, max=100)
result_Startparameter = dlg1.run() #returns a dictionary containing the entered values

dlg2 = gd.makeDialog('Kornklassen', "Kornklassen")
dlg2.addText("Geben Sie die Kornklassen in absteigender Korngroesse an!", 20, True)
dlg2.beginGroup("Korngroessen")
dlg2.addFloatInput("Korngroesse1", "Korngroesse 1 in mm", init=0)
dlg2.addFloatInput("Korngroesse2", "Korngroesse 2 in mm", init=0)
dlg2.addFloatInput("Korngroesse3", "Korngroesse 3 in mm", init=0)
dlg2.addFloatInput("Korngroesse4", "Korngroesse 4 in mm", init=0)
dlg2.addFloatInput("Korngroesse5", "Korngroesse 5 in mm", init=0)
dlg2.addFloatInput("Korngroesse6", "Korngroesse 6 in mm", init=0)
dlg2.addFloatInput("Korngroesse7", "Korngroesse 7 in mm", init=0)
dlg2.addFloatInput("Korngroesse8", "Korngroesse 8 in mm", init=0)
dlg2.endGroup()

dlg2.beginGroup("Verteilung")
dlg2.addFloatInput("Anteil1", "Aneil der Korngroesse 1 in %", init=0)
dlg2.addFloatInput("Anteil2", "Aneil der Korngroesse 2 in %", init=0)
dlg2.addFloatInput("Anteil3", "Aneil der Korngroesse 3 in %", init=0)
dlg2.addFloatInput("Anteil4", "Aneil der Korngroesse 4 in %", init=0)
dlg2.addFloatInput("Anteil5", "Aneil der Korngroesse 5 in %", init=0)
dlg2.addFloatInput("Anteil6", "Aneil der Korngroesse 6 in %", init=0)
dlg2.addFloatInput("Anteil7", "Aneil der Korngroesse 7 in %", init=0)
dlg2.addFloatInput("Anteil8", "Aneil der Korngroesse 8 in %", init=0)
dlg2.endGroup()
```

```

result_Verteilung = dlg2.run() #puts the input into a dictionary

Verteilung_tuple = sorted(result_Verteilung.items()) #changes the arbitrary keys of the dictionary into a sorted list of tuples
# the sorted list looks like this then: [('Anteil1', 40.0), ('Anteil2', 20.0), ..., ('Korngroesse1', 8.0), ('Korngroesse2', 7.0),...]
#first comes the sorted 'Anteil' and then the sorted 'Korngroesse' but it's all in one list - so we have to divide it into two separate
lists

Werte = [el[1] for el in Verteilung_tuple] #this function picks out only the value inside the tuples and puts them into the list "Werte"
Werte_Anteil=Werte[:len(Werte)/2] #puts all values of 'Anteile' into a sorted list
Werte_Korngr=Werte[len(Werte)/2:] #puts all values of 'Korngroesse' into a sorted list

Poro = result_Startparameter.items()[0][1] #assigns the value of the porosity to the variable 'Poro'

#Check if the sum of the entered shares values 100 or not - if it doesn't it will end the macro
if sum(Werte_Anteil) != 100:
    gd.msgBox('ERROR: Die Summe der Anteile ergibt nicht 100!')
    exit()

# ----- GeoDict:CreateProjectFolder -----
CreateProjectFolder_args = {
    'FolderName' : Project_Folder #Macro Packung erstellen Schleife'
}

gd.runCmd("GeoDict:CreateProjectFolder", CreateProjectFolder_args, Header['Release'])

textfile = open('Erzielte_Korngroessenanteile_in_der_Modellpackung.txt', 'w')
textfile.write('Korngroessenanteile\n\n')
textfile.write('Hier sind die erzielten Korngroessenanteile in der fertigen Packung aufgelistet, wobei die Anteile aufsummiert
werden! D.h. die nächstkleinere Korngroesse enthält auch jenen Anteil der höheren Korngroessen! Dieser Anteil muss zur
Ermittlung des eigentlichen Anteils abgezogen werden!\n\n' + 'Sollporosität = ' + str(Poro)+ '\n\n')
textfile.close()

Porofile =open('Porosität_Fortschritt.txt', 'w')
Porofile.write('Porosität\n\n')
Porofile.close()

#-----Modellpartikel (GAD-Files) laden und verändern mit Schleife-----

##! Sollten die Partikel schon fertig skaliert sein, muss diese Schleife beim nächsten Mal nicht noch einmal durchgegangen
werden!!
##!Dieser Abschnitt bis zur nächsten Hauptüberschrift "Einfügen der skalierten Modellpartikel" kann dann ausgeklammert werden!

f = glob.glob1('D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/', 'Modellpartikel*.gad') #scans the file for GAD-Data
with the name 'Modellpartikel' and puts them into the dictionary f - arbitrary order!

for file in sorted(f): #sorts the dictionary f and goes through the files in it

    filepath = "D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/" + file
    for x in Werte_Korngr:

        LoadGadFile_args = {
            'GADFileName' : filepath
        }
        gd.runCmd("GeoDict:LoadGadFile", LoadGadFile_args, Header['Release']) #load File

        EditDomain_args = {
            'Domain' : {
                'PeriodicX' : True,
                'PeriodicY' : True,
                'PeriodicZ' : True,
                'OriginX' : -0.00015,
                'OriginY' : -0.00015,
                'OriginZ' : -0.00015,
                'VoxelLength' : 5e-05,
                'DomainMode' : 'VoxelNumber',
                'NX' : 560,
                'NY' : 600,
                'NZ' : 1000,
                'Material' : {
                    'Name' : 'Air',
                    'Type' : 'Fluid',
                    'Information' : ""
                },
                'OverlapMode' : 'GivenMaterial',
                'OverlapMaterialID' : 2,
                'HollowMaterialID' : 0
            }
        }

```

```

    }

    gd.runCmd("GadGeo:EditDomain", EditDomain_args, Header["Release"])

    #scaling range for size of particle in a class
    range=x*0.8+x*0.4*random.random()

    ScaleGad_args = {
        'ScalingFactor' : range/10
    }
    gd.runCmd("GadGeo:ScaleGad", ScaleGad_args, Header["Release"]) #Scale File

    Reassign_args = {
        'OldMaterialID' : 1,
        'NewMaterialID' : 3
    }

    gd.runCmd("ProcessGeo:Reassign", Reassign_args, Header["Release"]) #Reassign MaterialID

    index = filepath.find('.gad')
    new_filename = filepath[:index] + '_Scaled_' + str(x) + filepath[index:]

    SaveFile_args = {
        'FileName': new_filename
    }
    gd.runCmd("GeoDict:SaveFile", SaveFile_args, Header["Release"]) #Save File

#-----Einfügen der skalierten Modellpartikel-----

#Da das Einfügen der Modellpartikel nur in eine bereits geöffnete Struktur erfolgen kann, muss zuerst ein "Startpartikel" gewählt
werden.

startfile = glob.glob1('D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/', "Modellpartikel_1_S*")[-1] #searches for all
scaled particles 1, puts them in a list and gives back the largest scaled particle
path_startfile = "D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/" + str(startfile)

LoadGadFile_args = {
    'GADFileName' : path_startfile
}
gd.runCmd("GeoDict:LoadGadFile", LoadGadFile_args, Header["Release"])

Reassign_args = {
    'OldMaterialID' : 3,
    'NewMaterialID' : 1
}
gd.runCmd("ProcessGeo:Reassign", Reassign_args, Header["Release"]) #reassigns the MaterialID of the particle to distinguish
between a set particle and a new one (maybe overlapping one)

Part = 0.0

for x in Werte_Korng:

    # The percentage of each class will be put into a textfile
    if Werte_Korng.index(x) != 0:
        addtext = open('Erzielte_Korngroessenanteile_in_der_Modellpackung.txt', 'a')
        line = 'Korngroesse ' + str(Werte_Korng[Werte_Korng.index(x)-1]) + ' ' + 'Anteil ' + str(Part) + ' Partikelanzahl
' + str(Nr_of_Particles) + '\n'
        addtext.write(line)
        addtext.close()

        saved_file = str(gd.getProjectFolder()) + '/' + 'Packung mit ' + str(Werte_Korng.index(x)) + 'Kornklassen.gdt'

        SaveFile_args = {
            'FileName': saved_file
        }
        gd.runCmd("GeoDict:SaveFile", SaveFile_args, Header["Release"]) #Save File

    index = Werte_Korng.index(x)
    Sum_Anteil = sum(Werte_Anteil[:index+1])

    scaled_files = glob.glob1('D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/', "*Scaled_" + str(x) + "*.gad")

    Nr_of_Particles = 0

    while Part <= ((100-Poro)*Sum_Anteil)/100: #as long as the desired percentage is not accomplished more particles will
be added into the geometry

```

```

random_file = random.choice(scaled_files) #es sollen willkürlich Files geladen werden

AddGadFile_args = {
    'GADFileName' : ["D:/_AndreaHoeftberger\Arbitrary Packing of designed particles/" + str(random_file)],
    'Psi' : random.uniform(0,360), #Grad
    'Theta' : random.uniform(0,360),
    'Phi' : random.uniform(0,360),
    'ShiftX' : random.uniform(0,0.02), #Meter
    'ShiftY' : random.uniform(0,0.02),
    'ShiftZ' : random.uniform(0,0.035)
}
gd.runCmd("GadGeo:AddGadFile", AddGadFile_args, Header['Release']) #the particle is loaded into the
structure at a random place, rotated randomley

MaterialID_list = gd.getVoxelCounts3D() #16-element list of voxel counts for each color (material index) for
the currently loaded geometry

Add_Current_Poro_to_File = open('Porosität_Fortschritt.txt','a')
Add_Current_Poro_to_File.write(str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) + ' ' +
str((gd.getVoxelCounts3D()[0])*100.0/(560.0*600.0*1000.0))+'\n')
Add_Current_Poro_to_File.close()

if MaterialID_list[2] > 0: #Material ID 02 is the material of the Overlap

    Reassign_args = {
        'OldMaterialID' : 3,
        'NewMaterialID' : 0
    }
    gd.runCmd("ProcessGeo:Reassign", Reassign_args, Header['Release'])

    Reassign_args = {
        'OldMaterialID' : 2,
        'NewMaterialID' : 1
    }
    gd.runCmd("ProcessGeo:Reassign", Reassign_args, Header['Release'])

    #The function Delete is only applicable to a GAD-File (in Memory = green dot), but since the
structure is modified during the process the GAD-Data does not match the voxel geometry anymore
    # instead the inserted particle will be removed by reassigning its Material ID
    # DeleteGad_args = {
        # 'DeleteType' : False,
        # 'DeleteMaterial' : False,
        # 'DeleteMaterialID' : True,
        # 'Type' : 'Undefined',
        # 'MaterialID' : 3,
        # 'Material' : {
            # 'Type' : 'Undefined'
        # }
    # }
    # gd.runCmd("GadGeo>DeleteGad", DeleteGad_args, Header['Release'])

    continue

    Reassign_args = {
        'OldMaterialID' : 3,
        'NewMaterialID' : 1
    }
    gd.runCmd("ProcessGeo:Reassign", Reassign_args, Header['Release'])

    Nr_of_Particles = Nr_of_Particles + 1

    i = gd.getVoxelCounts3D()[1] #puts the value (=number of voxels) of the Material ID 1 into the variable i
    Part = (i/(560.0*600.0*1000.0))*100.0 #400x400x700 ist das "Domain" - entspricht der gesamten Voxelanzahl

# In the end the completed file has to be saved and the endporosity written into the textfile
i = gd.getVoxelCounts3D()[1] #puts the value (=number of voxels) of the Material ID 1 into the variable i
Part = (i/(560.0*600.0*1000.0))*100.0 #400x400x700 ist das "Domain" - entspricht der gesamten Voxelanzahl

addtext = open('Erzielte_Korngrößenanteile_in_der_Modellpackung.txt','a')
line = 'Korngrösse ' + str(Werte_Korngr[-1]) + ' ' + 'Anteil ' + str(Part) + '\n'
addtext.write(line)
addtext.close()

SaveFile_args = {
'FileName': gd.getProjectFolder() + '/' + 'Fertige Packung.gdt'
}
gd.runCmd("GeoDict:SaveFile", SaveFile_args, Header['Release']) #Save File

```


Parameter-Datei des Modells "Polyeder-36"
Dateiname: Parameter.txt

Domain [Voxel]:
450 450 802
Solid Share [%] Deviation [%]:
44.136572 20
Particle size analysis [mm] [%]:
15 0
8.15 17.19
5.15 23.89
3.0 35.04
1.5 21.71
0.75 2.14
0.4275 0.02
0.3025 0.01
0.125 0.01

**Exemplarischer Ausschnitt der Ein-/Ausgabe-Oberfläche während der
Packungsgenerierung mit "PackageCreator"**

```
Total solid share: 38.9683%
Particle count: 6777
Solid voxel: 63286468
Empty voxel: 99118532
Particle ID: 6   Real scale: 11.4947% = 1.72421mm           Volume: 887
Nr. of tries: 5
Basic scale (1.5) up to 43.1788% => 70124540 Voxel
Current runtime: 03:53:17
time to add: 0.117913sec

Total solid share: 38.9689%
Particle count: 6778
Solid voxel: 63287391
Empty voxel: 99117609
Particle ID: 12  Real scale: 11.3667% = 1.70501mm           Volume: 923
Nr. of tries: 1
Basic scale (1.5) up to 43.1788% => 70124540 Voxel
Current runtime: 03:53:17
time to add: 0.086531sec

Total solid share: 38.9695%
Particle count: 6779
Solid voxel: 63288389
Empty voxel: 99116611
Particle ID: 3   Real scale: 9.33275% = 1.39991mm           Volume: 998
Nr. of tries: 8
Basic scale (1.5) up to 43.1788% => 70124540 Voxel
Current runtime: 03:53:17
time to add: 0.238697sec
```

Exemplarische Darstellung einer kleinen Box mit einem Korn in ASCII-Format
Dateiname: Particle_30.leS

```
3 5 9
000000000
000010000
000111000
000010000
000000000
000000000
000111000
001111100
000111000
000000000
000000000
000010000
000111000
000010000
000000000
```

Exemplarischer Ausschnitt aus der Statistik-Datei des Modells "Polyeder-36"

Dimension: { x450 y450 z802}
 Porosity: 55.859%
 Scale Diversity: 20%
 Solid Voxel: 71687160
 Number of particles: 19657
 Time to create: 07:51:10

Runtime	Basic_scale	Number_of_Particles	Solid_share	Solid_voxel
00:04:52	8.15	38	7.69867%	12503024
00:07:10	5.15	227	18.1672%	29504494
03:10:43	3	1851	33.6025%	54572220
06:54:17	1.5	8870	43.1795%	70125588
07:31:41	0.75	5406	44.1234%	71658547
07:32:58	0.4275	223	44.1322%	71672878
07:34:32	0.3025	258	44.1366%	71680040
07:50:32	0.125	2784	44.141%	71687160

Runtime	Basic_scale	Real_scale	Modelnumber	Volume
00:04:33	8.15	0.448091	1	343476
00:04:33	8.15	0.632683	10	19848
00:04:33	8.15	0.492956	13	326293
00:04:34	8.15	0.459532	5	435881
00:04:35	8.15	0.535081	14	229380
00:04:35	8.15	0.520859	9	450653
00:04:36	8.15	0.462375	15	304530
00:04:37	8.15	0.481116	17	222181
00:04:37	8.15	0.592647	14	310313
00:04:38	8.15	0.442829	5	390147
00:04:40	8.15	0.449714	7	689102
00:04:41	8.15	0.589592	9	650822
00:04:42	8.15	0.561422	13	482237
00:04:43	8.15	0.558539	7	1317809
00:04:44	8.15	0.463771	3	95240
00:04:44	8.15	0.465996	3	94782
00:04:44	8.15	0.531309	3	142968
00:04:44	8.15	0.472257	10	9445
00:04:44	8.15	0.572491	11	407207
00:04:44	8.15	0.537148	10	13047
00:04:45	8.15	0.649485	8	262693
00:04:46	8.15	0.49831	1	470793
00:04:46	8.15	0.507965	16	94085
00:04:46	8.15	0.633348	15	779057
00:04:47	8.15	0.624824	17	471200
00:04:47	8.15	0.602095	4	355448
00:04:47	8.15	0.537728	2	59660
00:04:48	8.15	0.520834	6	70336
00:04:48	8.15	0.614357	1	880789
00:04:49	8.15	0.448556	14	135758
00:04:49	8.15	0.447488	8	89274
00:04:49	8.15	0.503637	2	48972
00:04:49	8.15	0.621663	10	19074
00:04:50	8.15	0.568554	5	827291
00:04:50	8.15	0.639568	3	246096
00:04:51	8.15	0.496847	13	334151

Anleitung für "PackageCreator"

Generieren einer neuen Packung

- "PackageCreator" herunterladen unter: <https://github.com/Chris-3/PackageCreator> -> Ordner: „PackageCreator“ -> „PackageCreator.exe“
- Modellpartikelbibliothek als leS-Dateien in dasselbe Arbeitsverzeichnis laden
- Parameter.txt erstellen (Vorlage in Anhang V)
- „PackageCreator.exe“ durch Doppelklick ausführen
- Optionen hinzufügen oder ausschalten durch die Eingabe der entsprechenden Nummer und drücken der Eingabetaste
- Programmstart durch Eingabe von 0
- Nach erfolgreichem Beenden der Packungsgenerierung wird die fertige Packung als „new_package.leS“ in dem Arbeitsverzeichnis abgespeichert (wird bei einer bereits bestehenden Bezeichnung einfach überschrieben)
- Statistikdatei wird ebenfalls in das Arbeitsverzeichnis als „statistic.txt“ abgespeichert (wird bei einer bereits bestehenden Bezeichnung ebenfalls einfach überschrieben)
- Abbruch erfolgt durch: Strg + c
- Nach dem vorzeitigen Abbruch wird die Packung, welche bis zu diesem Zeitpunkt schon besteht, geladen

Anderes Arbeitsverzeichnis

- Sollte sich das Arbeitsverzeichnis mit den darin befindlichen Modellpartikeln und der Parameterdatei in einem anderen Ordner befinden, so kann dies in der Befehlszeile ausgewählt werden durch Eingabe von 6 -> dann Programm starten durch die Eingabe von 0 -> Eingabe des gesamten (absoluten) Pfades des neuen Arbeitsverzeichnisses

Laden einer bereits bestehenden Packung zum weiteren Einschichten

- Ersten 5 Schritte wie bei „Generieren einer neuen Packung“ ausführen
- Eingabe von 5 um Option „Load existing Package“ einzuschalten
- Eingabe von 0
- Dateiname der bereits bestehenden Packung eingeben (der Name muss nicht vollständig sein, sich aber eindeutig einer Datei zuordnen lassen!)
- Die Packung muss sich im selben Arbeitsverzeichnis finden, wie die Modellpartikelbibliothek und die Parameterdatei

Code des "PackageCreator" – Main: PackageCreator.cpp

```

1  #include <iostream>
2  #include <vector>
3  #include <map>
4  #include <algorithm>
5
6  enum opt_flags
7  {
8      SPIN = 1 << 0,
9      GRIT_COUNT = 1 << 1,
10     MORE_INFO = 1 << 2,
11     FILL_HOLES = 1 << 3,
12     COLOUR_ID = 1 << 4,
13     LOAD_PACKAGE = 1 << 5,
14     CURRENT_DIR = 1 << 6
15 };
16
17
18 #include "get_filenames.h"
19 #include "Grit.h"
20 #include "Package.h"
21
22 //this part handles user interrupt signals to make shure programm shuts
down with saving
23 sig_atomic_t stopFlag =0;
24 void handler(int)
25 {
26
27     cout <<"\n\n=====\n"
28         << "-----Interrupt detected!-----\n"
29         << " -> wait for saving current progress\n -> press strg + c a
second time for immediate shutdown"
30         << "\n=====\n\n"
31         ;
32     stopFlag = 1;
33 }
34
35
36 // this function searches for every file named "Particle_" and creates a
Grit object for each
37 bool getGritsofScale(std::vector<Grit> &, const std::vector<std::string>
&, const double&);
38
39 int main(int argc, char* argv[])
40 {
41     signal(SIGINT, handler);
42     int16_t options = SPIN + COLOUR_ID + CURRENT_DIR;
43
44     cout << "\n*****PackageCreator*****\n";
45     std::vector<std::string> fnames; //here are all filenames in target
directory saved
46
47     get_default_settings(fnames, options);
48
49     for (auto const & param : fnames)
50     {
51         if (param.find("Parameter") != std::string::npos)
52         {
53             Package pack(param,options,fnames.back()); //class Package is
initialised with parameter file

```

```
54
55         std::vector<Grit> x;//this will be the list of the original
particles
56
57         if (!getGritsofScale(x, fnames, pack.get_scale()->first))
58             {
59                 std::cout << " No file with name \"Particle_\" found\n
press any key to continue\n";
60                 std::cin.get();
61                 return 0;
62             }
63
64         std::cout << "\n Start adding particles: \n";
65         pack.fill_package(x);
66         return 0;
67     }
68 }
69     std::cout << "No file with name \"Parameter\" found\n press any key
to continue\n";
70     std::cin.get();
71     return 0;
72 }
73
74
75 //this function fills list with particle data (returns false when no
particle files)
76 bool getGritsofScale(std::vector<Grit> &x, std::vector<std::string>
const& fnames, const double& sc)
77 {
78     bool no_file = false;
79     for (unsigned int i = 0; i < fnames.size(); i++)
80     {
81         if (fnames[i].rfind("Particle_") != std::string::npos)
82             {
83                 Grit n(fnames[i]);
84                 x.push_back(n);
85                 no_file = true;
86             }
87     }
88 }
89     return no_file;
90 }
```

get_filenames.h

```

1  #pragma once
2  #include <boost/filesystem.hpp>
3
4  using std::cout;
5  using std::string;
6  using namespace boost::filesystem;
7
8
9
10
11 //this function searches targeted directory and saves all filenames in
fnames
12 bool get_filenames(std::vector<std::string> &fnames, int16_t & options)
13 {
14     path p;
15     try
16     {
17         if (options&CURRENT_DIR)
18         {
19             p = current_path();
20         }
21         if (!(options&CURRENT_DIR))
22         {
23             cout << "\n Enter working directory: \n";
24             std::cin >> p;
25         }
26         if (exists(p))
27         {
28             if (is_regular_file(p))
29                 cout << p << " size is " << file_size(p) << '\n';
30
31             else if (is_directory(p))
32             {
33                 for (auto&& x : directory_iterator(p))
34                 {
35                     std::string n = x.path().generic_string();
36                     fnames.push_back(n);
37                 }
38                 if (options&LOAD_PACKAGE)
39                 {
40                     while (1)
41                     {
42                         int old_size = fnames.size();
43                         std::string str;
44                         cout << "Enter filename of package to load
(only files in working directory can be loaded):\n";
45                         std::cin >> str;
46                         for (auto const& n : fnames)
47                         {
48                             if (n.find(str) != std::string::npos)
49                             {
50                                 fnames.push_back(n);
51                                 break;
52                             }
53                         }
54                         if (old_size != fnames.size())break;
55
56                         cout << "\n No file named \"" << str << "\"
found!\n";
57

```

```

58             }
59         }
60         return true;
61     }
62     else
63         cout << p << " exists, but is not a regular file or
directory\n";
64         return false;
65     }
66     else
67         cout << p << " does not exist\n";
68         return false;
69     }
70
71     catch (const filesystem_error& ex)
72     {
73         cout << ex.what() << "\n";
74         return false;
75     }
76 }
77
78 //this function prints a menu to get the new default settings
79 void get_default_settings(std::vector<std::string>& fnames, int16_t
&options)
80 {
81     //OPT.COLOUR_ID = false;
82     char choice = ' ';
83     std::string dir;
84     for (;;)
85     {
86
87         cout << "\n 0 Start programm "
88             << "\n 1 Spinning of particles: " << ((options&SPIN) ? "on"
: "off")
89             << "\n 2 Count particles: " << ((options&GRIT_COUNT) ? "on"
: "off")
90             << "\n 3 Get more info in statistic file: " <<
((options&MORE_INFO) ? "on" : "off")
91             << "\n 4 Assign every basic scale a diffent ID: " <<
((options&COLOUR_ID) ? "on" : "off")
92             << "\n 5 Load existing Package: " << ((options&LOAD_PACKAGE)
? "on" : "off")
93             << "\n 6 Change working directory: " << ((options&CURRENT_DIR)
? "off" : "on")
94             //<< "\n 7 Fill holes when overlapping voxels occur: " <<
(FILL_HOLES ? "on" : "off")
95             << "\n\n To alter default settings press corresponding
number, enter 0 to start program: "
96             ;
97
98         std::cin >> choice;
99
100        switch (choice)
101        {
102        case '1':
103            options ^= SPIN;
104            continue;
105        case '2':
106            options ^= GRIT_COUNT;
107            continue;
108        case '3':
109            options ^= MORE_INFO;
110        }

```

```
111     case '4':
112         options ^= COLOUR_ID;
113         continue;
114     case '5':
115         options ^= LOAD_PACKAGE;
116         continue;
117     case '6':
118         options ^= CURRENT_DIR;
119         continue;
120     /*case '7':
121         FILL_HOLES = !FILL_HOLES;
122         continue;*/
123     default:
124         while (!get_filenames(fnames, options));
125
126         return;
127     }
128 }
129 }
```

Grit.h

```

1  #pragma once
2  #include <iostream>
3  #include <vector>
4  #include <forward_list>
5  #include <fstream>
6  #include <string>
7  #include <stdlib.h>      /* srand, rand */
8  #include <iomanip>      // std::setprecision
9  #include <math.h>
10 #include <typeinfo>
11 #include <unordered_set>
12
13 # define M_PI           3.14159265358979323846 /* pi */
14
15 the following struct is the implementaion of a mathematical 3D vector
class
16
17 *****
***
18 template <typename T>
19 struct coordinate {
20
21     T x;
22     T y;
23     T z;
24
25     coordinate<T> normalized() const
26     {
27         T n = 0;
28         coordinate<T> temp = *this;
29         n = sqrt(x*x + y * y + z * z);
30         return temp / n;
31     }
32
33     coordinate<T> cross_product(coordinate<T> const& v) const
34     {
35         return { y*v.z - z * v.y,
36                 z*v.x - x * v.z,
37                 x*v.y - y * v.x };
38     }
39
40
41 };
42 template <typename T1, typename T2>
43 inline coordinate<T2> operator + (coordinate<T1> const & a,
coordinate<T2> const & b)
44 {
45     return { a.x + b.x ,a.y + b.y ,a.z + b.z };
46 }
47 template <typename T1, typename T2>
48 inline coordinate<T2> operator - (coordinate<T1> const & a,
coordinate<T2> const & b)
49 {
50     return { a.x - b.x ,a.y - b.y ,a.z - b.z };
51 }
52 template <typename T1, typename T2>
53 inline coordinate<T2> operator * (coordinate<T1> const & a, T2 const &

```

```

b)
54 {
55     return { a.x*b, a.y*b, a.z*b };
56 }
57 template <typename T1, typename T2>
58 inline coordinate<T2> operator / (coordinate<T1> const & a, T2 const &
b)
59 {
60     return { a.x / b, a.y / b, a.z / b };
61 }
62 template <typename T1, typename T2>
63 inline T1 operator * (coordinate<T1> const & a, coordinate<T2> const &
b)
64 {
65     return (a.x * b.x + a.y * b.y + a.z * b.z);
66 }
67 template <typename T>
68 inline bool operator == (coordinate<T> const & a, coordinate<T> const &
b)
69 {
70     if(a.x != b.x)return false;
71     if(a.y != b.y)return false;
72     if(a.z != b.z)return false;
73     return true;
74 }
75 template<class T>
76 std::ostream& operator<<(std::ostream &os, const coordinate<T>&v) {
77     os << "{ x" << v.x << " y" << v.y << " z" << v.z << " }";
78     return os;
79 }
80 template <typename T1, typename T2>
81 inline bool operator < (coordinate<T1> const & a, coordinate<T2> const
& b)
82 {
83     if(a.x < b.x)return true;
84     if (a.y < b.y)return true;
85     if (a.z < b.z)return true;
86     return false;
87 }
88 template <typename T1, typename T2>
89 inline bool operator > (coordinate<T1> const & a, coordinate<T2> const
& b)
90 {
91     if (a.x > b.x)return true;
92     if (a.y > b.y)return true;
93     if (a.z > b.z)return true;
94     return false;
95 }
96 template <typename T>
97 inline coordinate<T> abs(const coordinate<T> & a)
98 {
99     return{ abs(a.x),abs(a.y),abs(a.z) };
100 }
101
102
103 ****
104 the following class manages the initialisation of Particles and
105 documents the parameters of scaled and spinned particles
106 ****
107 **/
108
109
110

```

```
107
108 class Grit
109 {
110 public:
111     Grit();
112     Grit(std::string const&);
113     ~Grit();
114
115
116     void is_frame(const coordinate<int> &);
117     //checks if current particle is one of the outer most
118     void center_of_gravity();
119     //calculates center of gravity
120     void convert_coordinates();
121     //converts vectors to new starting point from center of gravity
122     void get_rot_param(const coordinate<double>&,const double&);
123     //this function saves the rotation parameters for every added
particle
124
125 private:
126
127     std::forward_list<coordinate<int>> p_img;
128     //saves a vector to every solid voxel found in the original file
129     coordinate<int> center_p;
130     //vector to calculated center of gravity from absolut starting point
131     coordinate<int> dim_part;
132     //dimension of the file the particle is loaded from
133     std::vector<coordinate<int>> frame_points;
134     //outer limits of the particle (original file)
135     std::vector<coordinate<int>> frame_points_new;
136     //outer limits of the particle (spinned and scaled)
137     uint32_t volume_or;
138     //original Volume of particle
139     uint32_t volume;
140     //volume after spinning and scaling
141     std::string id;
142     //modelnumber of particle
143     std::string grit_path;
144     //filepath of source file
145     coordinate<double> rot_v;
146     //parameter for rotation matrix (unit vector)
147     double rot_alpha;
148     //parameter for rotation matrix (alpha)
149     coordinate<int> v_trans;
150     //new center of gravity in package
151
152     void get_id(std::string const&);
153     //extracts Modelnumber from filename
154
155     friend class Package;
156 };
```

Grit.cpp

```

1  #include "Grit.h"
2  #define DEBUG 0
3  #define DEBUG1 0
4  #define DEBUG2 0
5
6  using std::cout;
7  Grit::Grit() {}
8
9
10 Grit::Grit(std::string const& d) :grit_path(d)
11 {
12
13     std::fstream file(grit_path, std::ios_base::in);
14     uint32_t count = 0;
15     bool voxl = 0;
16     bool hasbeenGrit = false;
17
18     std::cout << "\n Initialise Particle: " << grit_path << "\n";
19     if (file.is_open())
20     {
21         file >> dim_part.x >> dim_part.y >> dim_part.z;
22         while (file >> voxl)
23         {
24             if (voxl)
25             {
26                 //create vektor from file
27                 coordinate<int> x = { static_cast<int>(count /
(dim_part.z*dim_part.y)
28                 , static_cast<int>((count % (dim_part.z*dim_part.y))
/ dim_part.z)
29                 , static_cast<int>(count%dim_part.z) };
30                 p_img.push_front(x);
31                 volume++;
32                 hasbeenGrit = true;
33             }
34             if (hasbeenGrit && (((count / (dim_part.z*dim_part.y)) -
p_img.front().x) > 5))
35             {
36                 break;
37             }
38             count++;
39         }
40     }
41     file.close();
42
43     center_of_gravity();
44     get_id(grit_path);//extracts model number
45     frame_points.resize(6, center_p);
46     frame_points_new.resize(6, center_p);
47     if (DEBUG)
48     {
49         std::cout << "\n volume" << volume << "\n centerpoint:";
50         std::cout << "\n frame:";
51     }
52
53     convert_coordinates();//converts source point to center of gravity
for every vector
54     std::cout << " Particle ID: " << id << "\t Volume: " << volume<<"\n";
55
56 }

```

```
57
58 Grit::~Grit() {}
59
60
61 void Grit::is_frame(const coordinate<int> & i)
62 {
63     if (frame_points[0].x < i.x)frame_points[0] = i;
64     if (frame_points[1].x > i.x)frame_points[1] = i;
65     if (frame_points[2].y < i.y)frame_points[2] = i;
66     if (frame_points[3].y > i.y)frame_points[3] = i;
67     if (frame_points[4].z < i.z)frame_points[4] = i;
68     if (frame_points[5].z > i.z)frame_points[5] = i;
69 }
70
71
72 void Grit::center_of_gravity()
73 {
74     center_p = { 0,0,0 };
75     for (auto const& n : p_img)
76     {
77         center_p = center_p + n;
78     }
79     center_p.x = center_p.x / volume;
80     center_p.y = center_p.y / volume;
81     center_p.z = center_p.z / volume;
82 }
83
84 void Grit::convert_coordinates()
85 {
86     for (auto& n : p_img)
87     {
88         is_frame(n);
89         n = n - center_p;
90     }
91     for (auto& n : frame_points)
92     {
93         n = n - center_p;
94     }
95 }
96
97 void Grit::get_id(std::string const & d)
98 {
99     for (auto i = (d.find("Particle_") + 9); d[i] != '_'; i++)
100     {
101         if (d[i] == '.')break;
102         id.push_back(d[i]);
103     }
104 }
105
106 void Grit::get_rot_param(const coordinate<double>& _rot_v, const double
& _rot_alpha)
107 {
108     rot_v = _rot_v;
109     rot_alpha = _rot_alpha;
110 }
```

Package.h

```

1  #pragma once
2  #include <iostream>
3  #include <vector>
4  #include <list>
5  #include <string>
6  #include <sstream>
7  #include <map>
8  #include <ctime>
9  #include <ratio>
10 #include <chrono>
11 #include <signal.h>
12
13 #include "Grit.h"
14
15
16 using namespace std::chrono;
17 using std::cout;
18 extern sig_atomic_t stopFlag;
19
20
21
22 *****
23 ***
24 the following class initialises the package,
25 scales and spinnes the particles, adds them to the package
26 and documents the data
27 *****
28 **/
29
30 class Package
31 {
32 public:
33     //Constructor
34     Package(const std::string &, const int16_t &, const std::string &);
35     //Destructor
36     ~Package();
37
38     //this function fills the package with particles
39     void fill_package(std::vector<Grit>&);
40     //returns current basic scale and solid share border
41     std::map<double, uint32_t>::const_reverse_iterator get_scale();
42     //generates output file from calculated package
43     void create_file();
44     //prints the current status after adding particle/grit
45     void status(Grit const&);
46
47 private:
48     //options
49     const int16_t options;
50     //the data of this member will be saved in output file
51     std::vector<std::vector<std::vector<uint8_t>>> package;
52     //dimension of the created package
53     coordinate<int> dim_pack;
54     //path to source directory
55     std::string pack_path;
56     //total volume of package
57     uint32_t max_vol;

```

```

57 //number of solid voxels in package
58 uint32_t solid_vox;
59 //all scaling factors and corresponding solid share
60 std::map<double, uint32_t> por_threshold;
61 //current scaling factor and corresponding solid share
62 std::map<double, uint32_t>::const_reverse_iterator it_now;
63 //range for scale diversity in %
64 double scale_diversity;
65
66 //counter for added particles
67 uint32_t count;
68 //maximum solid share %
69 double max_solid;
70 //counter for number of tries for each particle
71 uint32_t tried;
72 //runtime of the programm
73 steady_clock::time_point run_t;
74 //Array for checking neighbouring voxels
75 const std::vector<coordinate<int>> v_fill = { {0,0,0}, {1,1,1}
, {0,1,1}, {0,0,1}, {0,1,0}, {1,0,0}, {1,1,0}, {1,0,1} , { -1,-1,-1 }, { 0,-1,-1
}, { 0,0,-1 }, { 0,-1,0 }, { -1,0,0 }, { -1,-1,0 }, { -1,0,-1 }};
76 //counts voxels added per particle
77 int added_vox;
78 //the calculated real scale including scale diversity
79 double real_scale;
80 //record of every particle added
81 std::vector<std::string> stat;
82 //general statistic for every scaling factor
83 std::vector<std::string> stat_general;
84 //colour ID in final Package
85 uint8_t colour;
86 //number of different particlemodels
87 unsigned int nr_of_grit;
88 //current particlemodel
89 unsigned int grit_choice;
90
91
92 //checks for existing particles were new particle should be inserted
93 void check_if_free(Grit&, coordinate<int> const&);
94 //if all checks are valid this function inserts particle at given
position and orientation
95 void add_grit(Grit&, const coordinate<int> &);
96 //checks vector for periodicity
97 inline void is_in_frame(coordinate<int>&);
98 //spins the vector and scales it
99 inline void spin_and_scale(coordinate<int>&, const Grit&);
100 //creates a vektor pointing from point to center of gravity
101 inline coordinate<int> dir_to_center(coordinate<int>);
102 //if coordinates overlap this function checks neighbouring voxels
for free spots
103 inline void fill_holes(coordinate<int>&, const int &);
104 //creates line for statistic for each particle added to package
105 void get_stat(const std::ostream&, int&);
106 //initialises empty package
107 void init_empty_pack();
108 //loads eisting package from file
109 void load_pack(const std::string &);
110
111
112 };

```

Package.cpp

```

1  #include "Package.h"
2  #include "random.h"
3
4  #define DEBUG 0
5  #define DEBUG1 0
6
7  enum opt_flags
8  {
9      SPIN = 1 << 0,
10     GRIT_COUNT = 1 << 1,
11     MORE_INFO = 1 << 2,
12     FILL_HOLES = 1 << 3,
13     COLOUR_ID = 1 << 4,
14     LOAD_PACKAGE = 1 << 5,
15     CURRENT_DIR = 1 << 6
16 };
17
18
19 //this function generates a random int 3D vector
20 inline coordinate<int> Random_V_int(const coordinate<int> & n)
21 {
22     std::random_device rd;
23     pcg rand(rd);
24     std::uniform_int_distribution<> x(0, n.x - 1);
25     std::uniform_int_distribution<> y(0, n.y - 1);
26     std::uniform_int_distribution<> z(0, n.z - 1);
27
28                                     return {
static_cast<int>(x(rand)), static_cast<int>(y(rand)), static_cast<int>(z(rand)
) };
29 }
30
31 //this function generates a random double 3D vector
32 inline coordinate<double> Random_V_double(const coordinate<int> & n)
33 {
34     std::random_device rd;
35     pcg rand(rd);
36     std::uniform_real_distribution<> x(1 - n.x, n.x - 1);
37     std::uniform_real_distribution<> y(1 - n.y, n.y - 1);
38     std::uniform_real_distribution<> z(1 - n.z, n.z - 1);
39
40     return { x(rand), y(rand), z(rand) };
41 }
42
43 // generates random Int
44 inline int Random_No(const unsigned int &n)
45 {
46     std::random_device rd;
47     pcg rand(rd);
48     std::uniform_int_distribution<> p(0, n);
49     return p(rand);
50 }
51
52 // generates random double
53 inline double Random_No(const double& n)
54 {
55     std::random_device rd;
56     pcg rand(rd);
57     std::uniform_real_distribution<> p(0, n);
58     return p(rand);

```

```

59  }
60
61  void Package::init_empty_pack()
62  {
63      double i = 0;
64      int perc = 10;
65      package.resize(dim_pack.x);
66      for (auto& y : package)
67      {
68          y.resize(dim_pack.y);
69          for (auto& z : y)
70          {
71              z.resize(dim_pack.z, 0);
72          }
73          if (perc < (i / dim_pack.x) * 100)
74          {
75              std::cout << "Initialise package: " << perc << "%\n";
76              perc += 10;
77          }
78          i++;
79      }
80  }
81  }
82
83  void Package::load_pack(const std::string & str)
84  {
85
86      std::fstream file_pack(str, std::ios_base::in);
87      coordinate<int> dim_pack_old;
88      file_pack >> dim_pack_old.x >> dim_pack_old.y >> dim_pack_old.z;
89      uint8_t vox1 = 0;
90      uint32_t count = 0;
91      uint16_t perc = 10;
92
93      cout << "\n loading package from: " << str << "\n";
94      while (file_pack >> vox1)
95      {
96          vox1 = vox1 - 48;
97          if (vox1)
98          {
99              coordinate<int> v = { static_cast<int>(count /
100 (dim_pack_old.z*dim_pack_old.y))
101                                     , static_cast<int>((count %
102 (dim_pack_old.z*dim_pack_old.y)) / dim_pack_old.z)
103                                     , static_cast<int>(count%dim_pack_old.z) };
104              if ((v.x < dim_pack.x) && (v.z < dim_pack.z) && (v.z <
105 dim_pack.z))
106              {
107                  package[v.x][v.y][v.z] = vox1;
108                  solid_vox++;
109              }
110              if (perc < ((count * 100) /
111 (dim_pack_old.x*dim_pack_old.z*dim_pack_old.y) ))
112              {
113                  std::cout << "Load package: " << perc << "%\n";
114                  perc += 10;
115              }
116              count++;
117          }
118      }
119  }
120  Package::Package(const std::string & d, const int16_t & opt, const

```

```

std::string & load_file) :pack_path(d), options(opt)
118 {
119
120
121     run_t = steady_clock::now();
122     std::fstream file(pack_path, std::ios_base::in);
123     double scale = 0.0, solid_share = 0.0;
124     uint32_t temp = 0;
125     std::string line;
126     std::istringstream iss;
127     tried = 0;
128     colour = 1;
129     grit_choice = 0;
130     //here Parameters are read from Parameter file
131     if (file.is_open())
132     {
133         std::getline(file, line);
134         std::getline(file, line);
135         iss.str(line);
136         iss >> dim_pack.x >> dim_pack.y >> dim_pack.z;
137         std::getline(file, line);
138         std::getline(file, line);
139         iss.clear();
140         iss.str(line);
141         iss >> max_solid >> scale_diversity;
142         max_vol = dim_pack.x*dim_pack.y*dim_pack.z;
143         std::getline(file, line);
144
145         //while (file >> scale >> solid_share)
146         while (std::getline(file, line))
147         {
148             iss.clear();
149             iss.str(line);
150             iss >> scale >> solid_share;
151             if (!(options&GRIT_COUNT)) temp = temp +
static_cast<uint32_t>(((solid_share*max_solid) / 10000)*max_vol);
152             if ((options&GRIT_COUNT)) temp = temp +
static_cast<uint32_t>(solid_share);
153             por_threshold.insert(std::pair<double, uint32_t>(scale,
temp));
154         }
155     }
156     file.close();
157     it_now = por_threshold.rbegin();
158
159     //here the 3D Array for the Package is created this variable will
be saved in output file
160
161
162     init_empty_pack();
163     if (options&LOAD_PACKAGE) load_pack(load_file);
164
165     //here is a general summery of the stats printed in command line
166     cout << "\n\n Package parameters:"
167         << "\n Package volume: " << max_vol
168         << "\n Solid share: " << max_solid << "%"
169         << "\t Scale diversity: " << scale_diversity << "%"
170         ;
171     //here are the scaling factors with according solid limits printed
172     for (auto it = por_threshold.rbegin(); it != por_threshold.rend();
++it)
173     {
174         cout << "\n Basic scale: " << it->first

```

```

175         << "\t=>\t Solid share border: " << static_cast<double>(it-
>second) / static_cast<double>(max_vol) * 100 << "%";
176     }
177     cout << "\n\n";
178
179 }
180
181 Package::~Package()
182 {
183     create_file();
184 }
185
186 //spins the vector and scales it
187 inline void Package::spin_and_scale(coordinate<int> &v, const Grit& p)
188 {
189     coordinate<double> x = v * static_cast<double>(1);
190     if (options&SPIN)
191     {
192         x = p.rot_v*(p.rot_v*x)*(1 - cos(p.rot_alpha))
193             + x * cos(p.rot_alpha)
194             + (p.rot_v.cross_product(x))*sin(p.rot_alpha);
195     }
196     x = x * real_scale;
197
198     static_cast<int>(x.x), static_cast<int>(x.y), static_cast<int>(x.z) };
199
200 //creates line for statistic for each particle added to package
201 inline void Package::fill_holes(coordinate<int>& n, const int &i)
202 {
203     n = { n.x + dir_to_center(n).x * v_fill[i].x,
204         n.y + dir_to_center(n).y * v_fill[i].y,
205         n.z + dir_to_center(n).z * v_fill[i].z };
206 }
207
208 //if all checks are valid this function inserts particle at given
position and orientation
209 void Package::add_grit(Grit& p_to_add, const coordinate<int> &v)
210 {
211     added_vox = 0;
212     for (auto n : p_to_add.p_img)
213     {
214         spin_and_scale(n, p_to_add);
215         n = n + v;
216         is_in_frame(n);
217         unsigned int fill;
218         if (options&FILL_HOLES) fill = v_fill.size();
219         if (!(options&FILL_HOLES)) fill = 1;
220         for (unsigned int i = 0; i < fill; i++)
221         {
222             fill_holes(n, i);
223             if (package[n.x][n.y][n.z])continue;
224             is_in_frame(n);
225             package[n.x][n.y][n.z] = colour;
226             solid_vox++;
227             added_vox++;
228             break;
229         }
230     }
231     count++;
232     status(p_to_add);
233     tried = 0;
234     grit_choice = Random_No(nr_of_grit);

```

```

235 }
236
237
238 //checks for existing particles were new particle should be inserted
239 void Package::check_if_free(Grit& p, const coordinate<int> & v)
240 {
241
242         p.get_rot_param(Random_V_double(dim_pack).normalized(),
Random_No(M_PI * 2));
243
244         real_scale = (it_now->first / por_threshold.rbegin()->first) +
((Random_No(scale_diversity * 2) / 100.0) - (scale_diversity / 100.0))*
(it_now->first / por_threshold.rbegin()->first);
245         p.v_trans = v;
246         int n_new = 0;
247
248         for (auto n : p.frame_points)
249         {
250                 spin_and_scale(n, p);
251                 p.frame_points_new[n_new++] = n;
252                 n = n + v;
253                 is_in_frame(n);
254                 if (package[n.x][n.y][n.z])return;
255         }
256         n_new = 0;
257
258         for (auto n : p.p_img)
259         {
260                 spin_and_scale(n, p);
261                 n = n + v;
262                 is_in_frame(n);
263
264                 if (package[n.x][n.y][n.z])return;
265         }
266         add_grit(p, v);
267         return;
268 }
269
270
271 //this function fills the package with particles
272 void Package::fill_package(std::vector<Grit>& particles)
273 {
274         coordinate<int> v;
275         nr_of_grit = particles.size() - 1;
276
277         while (true)
278         {
279                 if (it_now->second <= solid_vox && !(options&GRIT_COUNT))
++it_now;
280                 if (it_now->second <= count && (options&GRIT_COUNT)) ++it_now;
281                 if (it_now == por_threshold.rend() || stopFlag)return;
282                 tried++;
283                 if (!(tried % 100000))cout << "|";
284                 if (!(tried % 1000000))cout << "X";
285                 v = Random_V_int(dim_pack);
286                 if (package[v.x][v.y][v.z])continue;
287
288                 check_if_free(particles[grit_choice], v);
289         }
290 }
291
292 //returns current basic scale and solid share border
293 std::map<double, uint32_t>::const_reverse_iterator Package::get_scale()

```

```

294 {
295     return it_now;
296 }
297
298
299 //generates output file from calculated package
300 void Package::create_file()
301 {
302     double i = 0;
303     int perc = 5;
304     pack_path.replace(pack_path.begin() + pack_path.find_last_of('/') +
305 1, pack_path.end(), "new_package.leS");
306
307     std::cout << "\n write file in: " << pack_path << "\n";
308
309     std::ofstream file;
310     file.open(pack_path);
311     file << dim_pack.x << ' ' << dim_pack.y << ' ' << dim_pack.z;
312     for (auto const& x : package)
313     {
314         for (auto const& y : x)
315         {
316             file << '\n';
317             for (unsigned int z = 0; z < ((y.size() * 2) - 1); z++)
318             {
319                 if (!(z % 2))file << std::to_string(y[z / 2]);
320                 if ((z % 2))file << ' ';
321             }
322         }
323         if (perc < (i / dim_pack.x) * 100)
324         {
325             std::cout << "|";
326             perc += 5;
327         }
328         i++;
329     }
330     file.close();
331     std::cout << "\n";
332
333     steady_clock::time_point run_t_now = steady_clock::now();
334     duration<double> runtime = duration_cast<duration<double>>(run_t_now
335 - run_t);
336     int t = static_cast<int>(runtime.count());
337     std::ostringstream time;
338     time.flush();
339     time.fill('0');
340     time << std::setw(2) << t / 3600 << ":" << std::setw(2) << (t %
341 3600) / 60 << ":" << std::setw(2) << t % 60;
342
343     pack_path.replace(pack_path.begin() + pack_path.find_last_of('/') +
344 1, pack_path.end(), "statistic.txt");
345     std::cout << "\n write statistics in: " << pack_path << "\n";
346
347     file.open(pack_path);
348     file << " Dimension: \t" << dim_pack
349     << "\n Porosity: \t" << 100 - static_cast<double>(solid_vox) /
350     static_cast<double>(max_vol) * 100 << "%"
351     << "\n Scale Diversity: \t" << scale_diversity << "%"
352     << "\n Solid Voxel: \t" << solid_vox
353     << "\n Number of particles: \t" << count
354     << "\n Time to create: \t" << time.str()
355     << "\n\n";

```

```

352
353     file << "Runtime" << "\tBasic_scale" << "\tNumber_of_Particles" <<
"\tSolid_share" << "\tSolid_voxel" << "\n";
354
355     for (auto const& line : stat_general)
356     {
357         file << line;
358     }
359
360     file << "\n\nRuntime" << "\tBasic_scale" << "\tReal_scale" <<
"\tModelnumber" << "\tVolume" << "\n";
361
362     for (auto const& line : stat)
363     {
364         file << line;
365     }
366     file.close();
367
368     std::cout << "\n file schreiben fertig\n";
369     std::cin.get();
370 }
371
372 //creates line for statistic for each particle added to package
373 void Package::get_stat(const std::ostream& time, int&
count_p_scale)
374 {
375     std::ostream oss2;
376     oss2.flush();
377     oss2 << time.str()
378         << "\t" << it_now->first
379         << "\t" << count_p_scale
380         << "\t" << static_cast<double>(solid_vox) /
static_cast<double>(max_vol) * 100 << "%"
381         << "\t" << solid_vox << "\n";
382     stat_general.push_back(oss2.str());
383     count_p_scale = 0;
384     if (options&COLOUR_ID) colour++;
385     if (colour > 16) colour = 1;
386 }
387
388 //prints the current status after adding particle/grit
389 void Package::status(Grit const& p)
390 {
391     steady_clock::time_point run_t_now = steady_clock::now();
392     duration<double> runtime = duration_cast<duration<double>>(run_t_now
- run_t);
393     double t = runtime.count();
394     static double t_add = 0;
395     static int count_p_scale = 0;
396     std::ostream oss;
397     std::ostream time;
398     time.flush();
399     time.fill('0');
400     time << std::setw(2) << static_cast<int>(t) / 3600 << ":"
401         << std::setw(2) << (static_cast<int>(t) % 3600) / 60 << ":"
402         << std::setw(2) << static_cast<int>(t) % 60;
403     std::cout
404         << " Total solid share: " << static_cast<double>(solid_vox) /
static_cast<double>(max_vol) * 100 << "%"
405         << "\n Particle count: " << count
406         << "\n Solid voxel: " << solid_vox
407         << "\n Empty voxel: " << max_vol - solid_vox
408         << "\n Modelnumber: " << p.id << "\t Real scale: " << real_scale

```

```

* 100 << "% = " << por_threshold.rbegin()->first*real_scale << "mm \t Volume:
" << added_vox
409     << "\n Nr. of tries: " << tried
410     << "\n Basic scale (" << it_now->first << ") up to " <<
static_cast<double>(it_now->second) / static_cast<double>(max_vol) * 100 <<
"% =>" << it_now->second << " Voxel"
411     << "\n Current runtime: " << time.str()
412     << "\n time to add: " << t - t_add << "sec";
413     count_p_scale++;
414
415     if (!(options&GRIT_COUNT) && solid_vox >= it_now-
>second)get_stat(time, count_p_scale);
416     if ((options&GRIT_COUNT) && count >= it_now->second)get_stat(time,
count_p_scale);
417
418     oss.flush();
419     oss << time.str()
420     << "\t" << it_now->first
421     << "\t" << real_scale
422     << "\t" << p.id
423     << "\t" << added_vox
424     ;
425     if (options&MORE_INFO)
426     {
427         oss << "\t" << p.v_trans
428         << "\t" << p.rot_v
429         << "\t" << p.rot_alpha
430         ;
431         for (auto i : p.frame_points_new) oss << i;
432     }
433
434
435     oss << "\n";
436     stat.push_back(oss.str());
437     cout << "\n\n";
438     t_add = runtime.count();
439
440 }
441
442 //checks vector for periodicity
443 inline void Package::is_in_frame(coordinate<int> & i)
444 {
445     if (i.x < 0)i.x = dim_pack.x + i.x;
446     if (i.x >= dim_pack.x)i.x = i.x - dim_pack.x;
447
448     if (i.y < 0)i.y = dim_pack.y + i.y;
449     if (i.y >= dim_pack.y)i.y = i.y - dim_pack.y;
450
451     if (i.z < 0)i.z = dim_pack.z + i.z;
452     if (i.z >= dim_pack.z)i.z = i.z - dim_pack.z;
453 }
454
455 //creates a vektor pointing from point to center of gravity
456 inline coordinate<int> Package::dir_to_center(coordinate<int> v_temp)
457 {
458     if (abs(v_temp.x)) v_temp.x = v_temp.x / abs(v_temp.x);
459     if (abs(v_temp.y)) v_temp.y = v_temp.y / abs(v_temp.y);
460     if (abs(v_temp.z)) v_temp.z = v_temp.z / abs(v_temp.z);
461     return v_temp * static_cast<int>(-1);
462 }

```

random.h

```

1  /* Copyright (c) 2018 Arvid Gerstmann. */
2  /* This code is licensed under MIT license. */
3  #ifndef AG_RANDOM_H
4  #define AG_RANDOM_H
5  #include <stdio.h>
6  #include <random>
7
8  class splitmix
9  {
10 public:
11     using result_type = uint32_t;
12     static constexpr result_type(min)() { return 0; }
13     static constexpr result_type(max)() { return UINT32_MAX; }
14     friend bool operator==(splitmix const &, splitmix const &);
15     friend bool operator!=(splitmix const &, splitmix const &);
16
17     splitmix() : m_seed(1) {}
18     explicit splitmix(std::random_device &rd)
19     {
20         seed(rd);
21     }
22
23     void seed(std::random_device &rd)
24     {
25         m_seed = uint64_t(rd()) << 31 | uint64_t(rd());
26     }
27
28     result_type operator()()
29     {
30         uint64_t z = (m_seed += UINT64_C(0x9E3779B97F4A7C15));
31         z = (z ^ (z >> 30)) * UINT64_C(0xBF58476D1CE4E5B9);
32         z = (z ^ (z >> 27)) * UINT64_C(0x94D049BB133111EB);
33         return result_type((z ^ (z >> 31)) >> 31);
34     }
35
36     void discard(unsigned long long n)
37     {
38         for (unsigned long long i = 0; i < n; ++i)
39             operator()();
40     }
41
42 private:
43     uint64_t m_seed;
44 };
45
46 bool operator==(splitmix const &lhs, splitmix const &rhs)
47 {
48     return lhs.m_seed == rhs.m_seed;
49 }
50 bool operator!=(splitmix const &lhs, splitmix const &rhs)
51 {
52     return lhs.m_seed != rhs.m_seed;
53 }
54
55 class xorshift
56 {
57 public:
58     using result_type = uint32_t;
59     static constexpr result_type(min)() { return 0; }
60     static constexpr result_type(max)() { return UINT32_MAX; }

```

```

61     friend bool operator==(xorshift const &, xorshift const &);
62     friend bool operator!=(xorshift const &, xorshift const &);
63
64     xorshift() : m_seed(0xc1f651c67c62c6e0ull) {}
65     explicit xorshift(std::random_device &rd)
66     {
67         seed(rd);
68     }
69
70     void seed(std::random_device &rd)
71     {
72         m_seed = uint64_t(rd()) << 31 | uint64_t(rd());
73     }
74
75     result_type operator() ()
76     {
77         uint64_t result = m_seed * 0xd989bcacc137dcd5ull;
78         m_seed ^= m_seed >> 11;
79         m_seed ^= m_seed << 31;
80         m_seed ^= m_seed >> 18;
81         return uint32_t(result >> 32ull);
82     }
83
84     void discard(unsigned long long n)
85     {
86         for (unsigned long long i = 0; i < n; ++i)
87             operator() ();
88     }
89
90 private:
91     uint64_t m_seed;
92 };
93
94 bool operator==(xorshift const &lhs, xorshift const &rhs)
95 {
96     return lhs.m_seed == rhs.m_seed;
97 }
98 bool operator!=(xorshift const &lhs, xorshift const &rhs)
99 {
100    return lhs.m_seed != rhs.m_seed;
101 }
102
103 class pcg
104 {
105 public:
106     using result_type = uint32_t;
107     static constexpr result_type(min) () { return 0; }
108     static constexpr result_type(max) () { return UINT32_MAX; }
109     friend bool operator==(pcg const &, pcg const &);
110     friend bool operator!=(pcg const &, pcg const &);
111
112     pcg()
113         : m_state(0x853c49e6748fea9bULL)
114         , m_inc(0xda3e39cb94b95bdbULL)
115     {}
116     explicit pcg(std::random_device &rd)
117     {
118         seed(rd);
119     }
120
121     void seed(std::random_device &rd)
122     {
123         uint64_t s0 = uint64_t(rd()) << 31 | uint64_t(rd());

```

```
124     uint64_t s1 = uint64_t(rd()) << 31 | uint64_t(rd());
125
126     m_state = 0;
127     m_inc = (s1 << 1) | 1;
128     (void)operator() ();
129     m_state += s0;
130     (void)operator() ();
131 }
132
133 result_type operator() ()
134 {
135     uint64_t oldstate = m_state;
136     m_state = oldstate * 6364136223846793005ULL + m_inc;
137     uint32_t xorshifted = uint32_t(((oldstate >> 18u) ^ oldstate)
>> 27u);
138     int rot = oldstate >> 59u;
139     return (xorshifted >> rot) | (xorshifted << ((-rot) & 31));
140 }
141
142 void discard(unsigned long long n)
143 {
144     for (unsigned long long i = 0; i < n; ++i)
145         operator() ();
146 }
147
148 private:
149     uint64_t m_state;
150     uint64_t m_inc;
151 };
152
153 bool operator==(pcg const &lhs, pcg const &rhs)
154 {
155     return lhs.m_state == rhs.m_state
156         && lhs.m_inc == rhs.m_inc;
157 }
158 bool operator!=(pcg const &lhs, pcg const &rhs)
159 {
160     return lhs.m_state != rhs.m_state
161         || lhs.m_inc != rhs.m_inc;
162 }
163
164 #endif /* AG_RANDOM_H */
```