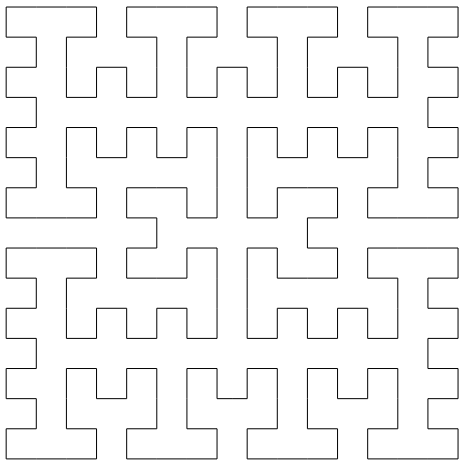
Anwendung diskreter raumfüllender Kurven

Graphpartitionierung und Kontaktsuche in der Finite-Elemente-Simulation

Jens-Michael Wierum



Dissertation

Schriftliche Arbeit zur Erlangung des Grades
eines Doktors der Naturwissenschaften

Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

Paderborn, Oktober 2003

Für Jörn, Maren
und Sandra

Danksagung

Mein besonderer Dank gilt meinem Betreuer Prof. Dr. Burkhard Monien für die Unterstützung meiner wissenschaftlichen Arbeiten am PC^2 in den vergangenen Jahren. Desweiteren bedanke ich mich bei Prof. Dr. Wilfried Hauenschild für die Begutachtung der Arbeit.

An dieser Stelle sind auch weitere Personen zu erwähnen, die meine Arbeit beeinflusst haben. Zu diesen zählen Dr. Lars Taenzer, Dr. Ralf Diekmann und weitere Mitarbeiter der Hilti AG, die mich für die Zusammenarbeit in verschiedenen Projekten begeistert haben, in denen Kenntnisse unterschiedlicher wissenschaftlicher Disziplinen zum Tragen kommen. Insbesondere die Fähigkeit zur Kommunikation im interdisziplinären Umfeld verdanke ich dieser Kooperation.

Mein Dank gilt allen Mitarbeitern des PC^2 für die langjährige Zusammenarbeit und die freundschaftliche Arbeitsatmosphäre. Die intensiven Diskussionen mit Dr. Jens Simon, Jan Hungershöfer, Achim Streit und Stefan Schamberger haben wesentlich zum Zustandekommen dieser Arbeit beigetragen. Weiterhin bedanke ich mich bei Prof. Dr. Odej Kao, Axel Keller, Oliver Marquardt und Kerstin Wielage, deren Kommentare zu Teilen der schriftlichen Arbeit für mich hilfreich waren.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Vor über hundert Jahren hat Mathematiker die Frage beschäftigt, ob eine stetige bijektive Abbildung des Intervalls $[0, 1]$ in das Einheitsquadrat $[0, 1]^2$ existiert. Im Jahre 1879 hat Eugen Netto gezeigt, dass eine solche bijektive Abbildung nicht stetig sein kann [54]. Damit reduzierte sich die Suche auf eine stetige surjektive Abbildung. Die erste Abbildung mit den gewünschten Eigenschaften zeigte Giuseppe Peano 1890 [63]. Die erste graphische Darstellung einer solchen Abbildung wurde ein Jahr später von David Hilbert angegeben [38]. Diese graphische Form der Definition führte zu der Bezeichnung der *raumfüllenden Kurve (space-filling curve)*. Im Laufe der Zeit wurde eine Vielzahl an raumfüllenden Kurven entwickelt. Die meisten sind für den zweidimensionalen Raum definiert, einige lassen sich jedoch auf Räume von beliebiger Dimension erweitern. Diese Arbeit konzentriert sich auf diskrete raumfüllende Kurven, welche Abbildungen zwischen diskreten Räumen darstellen. Diese ergeben sich aus allgemeinen raumfüllenden Kurven, wenn lediglich eine endliche Verfeinerung der rekursiven Definition angewendet wird. Der Schritt von kontinuierlichen Mengen auf diskrete ergibt in diesem Falle bijektive Abbildungen, so dass entsprechende Umkehrfunktionen definiert sind.

Die „lokalitätserhaltenden“ Eigenschaften diskreter raumfüllender Kurven werden in verschiedenen Anwendungsgebieten genutzt. Dabei ist die Definition der Lokalität von der Art der Anwendung abhängig. In einigen Bereichen ist eine Abbildung einer eindimensionalen Struktur auf ein mehrdimensionales Gitter gefordert. Diese Aufgabe ergibt sich z. B. bei parallelen Berechnungen auf Systemen, deren Kommunikationsstruktur dem Gitter entspricht [43, 71]. In diesem Falle sollen Elemente der eindimensionalen Struktur über möglichst kurze Wege innerhalb des Netzwerkes des parallelen Rechnersystems kommunizieren können.

Die Umkehrfunktion einer diskreten raumfüllenden Kurve erlaubt die Überführung mehrdimensionaler Strukturen in eine eindimensionale Form. Eine bedeutende Anwendung ist die Ordnung mehrdimensionaler Datenbanken [41, 53, 60]. Eine typische Anfrage erfolgt über die Angabe von Intervallen für jede der gespeicherten Dimensionen. Geometrisch ergibt sich eine Anfrageregion in Form eines r-dimensionalen Quaders, deren enthaltene Punktmenge zu bestimmen ist. Die Beantwortung einer solchen Anfrage

kann dann besonders schnell erfolgen, wenn die Punkte innerhalb des Quaders in wenigen zusammenhängenden Bereichen der eindimensionalen Struktur liegen. Damit ergibt sich eine hohe Lokalität, wenn die Nachbarschaft innerhalb des r-dimensionalen Raumes bei der Übertragung in die eindimensionale Struktur weitgehend erhalten bleibt. Zwei weitere Anwendungsgebiete in denen die Transformation in den eindimensionalen Raum eingesetzt werden kann, sind die Graphpartitionierung und die Kontaktsuche im Bereich der Simulation nach der Finite-Elemente-Methode. Die Eignung verschiedener raumfüllender Kurven für diese Einsatzgebiete wird innerhalb dieser Arbeit analysiert.

Der erste Teil der Arbeit analysiert die Graphpartitionierung über die Struktur raumfüllender Kurven. Ihre Aufgabe besteht in der Aufteilung der Knoten eines Graphen in eine gegebene Anzahl gleichgroßer Teile. Gleichzeitig soll die Anzahl der Kanten zwischen Knoten unterschiedlicher Partitionen minimiert werden. Die Fragestellung ergibt sich aus der geeigneten Aufteilung einer Berechnung in parallele Prozesse. Die Rechenlasten werden gleichmäßig auf die Recheneinheiten verteilt. Dabei sollen die Daten so auf dem Zielsystem verteilt werden, dass möglichst viele Operationen auf lokal gespeicherten Daten stattfinden und nur wenige Zugriffe auf fremden Speicher erfolgen. Für die NP-schwere Aufgabe der Graphpartitionierung sind viele Heuristiken entwickelt worden, die in vielen Untersuchungen verglichen worden sind [67, 73]. Die derzeit besten Methoden gehören zur Klasse der mehrstufigen Partitionierungsverfahren. Neben einer guten Qualität der Partitionierung sind sie ausreichend schnell, wenn eine statische Partitionierung gefordert ist.

Bei einigen parallelen Anwendungen ergibt sich eine stark schwankende Rechenlast in den Partitionen. Um eine hohe parallele Effizienz zu erzielen, muss ein ständiger Lastausgleich erfolgen. Damit ist eine schnelle Repartitionierung erforderlich, die gleichzeitig die Anzahl der externen Datenzugriffe gering hält. Aus dieser Fragestellung haben wir eine neue Form der Netzpartitionierung entwickelt, die wir als *implizite Partitionierung* bezeichnen. Über die Knoten des Graphen wird eine Indizierung durchgeführt. Jede Aufteilung dieser Indizierung in Intervalle ergibt eine Partitionierung, deren Qualität von dem gewählten Indizierungsschema abhängig ist. In dieser Arbeit wird die Indizierung über diskrete raumfüllende Kurven vorgestellt, die eine Ordnung auf Basis der geometrischen Positionen der Knoten des Graphen durchführt. Diese geometrischen Positionen sind i. A. bei Finite-Elemente-Netzen gegeben, da sie eine Diskretisierung des Raumes darstellen. Es werden analytische Ergebnisse zur Qualität der Partitionierung des Gitters über verschiedene raumfüllende Kurven hergeleitet. Dabei wird sowohl der schlechteste als auch der mittlere Fall untersucht. Desweiteren werden die Partitionierungseigenschaften der raumfüllenden Kurven in unstrukturierten Netzen experimentell untersucht. Die erzielten Partitionierungen werden anhand einer Menge von zwei- und dreidimensionalen Finite-Elemente-Netzen mit den Ergebnissen der Partitionierungsbibliothek Metis [45] verglichen.

Das zweite untersuchte Anwendungsgebiet ist die effiziente Kontaktsuche in Simulationsumgebungen. Die Aufgabe der Kontaktsuche besteht in der Bestimmung von Kollisionen zwischen Objekten im Raum, deren Bewegung simuliert wird. Die meisten

Anwendungsbereiche finden sich im Gebiet der Virtual Reality, bei der eine (zukünftige) Umgebung im Raum simuliert und visualisiert wird [32, 34, 84]. Insbesondere bei der Konstruktion von Bauteilen und deren Fertigungs- und Montageprozessen spielt die Kollisionserkennung eine entscheidende Rolle. Schon vor der Erstellung eines Prototyps sollte die Frage beantwortet sein, ob ein bestimmtes Bauteil problemlos an der vorgesehen Stelle integriert werden kann. Ähnliches gilt für die Bestimmung von Wegen für Roboter zwischen zwei Punkten ihres Einsatzgebietes, bei denen eine Kollision vermieden werden muss. Für diese und ähnliche Aufgaben sind viele geeignete Methoden der Kontaktsuche vorgeschlagen und analysiert worden [17, 30, 77]. Die meisten basieren auf hierarchischen Strukturen, über welche die Objekte im Raum geordnet werden.

Innerhalb der Simulation strukturmechanischer Prozesse mit Hilfe der Finite-Elemente-Methoden erfordert die Kontaktsuche oftmals einen sehr hohen Anteil an der Gesamtrechenzeit. In diesem Anwendungsbereich haben sich zellbasierte Methoden als effizientere Alternative erwiesen [26]. In dieser Arbeit wird eine neue Variante für eine zellbasierte Methode vorgestellt, die auf der Struktur raumfüllender Kurven basiert. Der daraus folgende Algorithmus wird hinsichtlich seiner Kosten in einem Szenario untersucht, welches die Anforderungen einer Kontaktsuche innerhalb der Simulation strukturmechanischer Prozesse abstrahiert. Aus der Kostenanalyse ergibt sich ein neues Lokalitätsmaß für raumfüllende Kurven, welches die Auswirkungen der Eigenschaften der Kurven im betrachteten Einsatzgebiet beschreibt. Eine Analyse verschiedener Kurven im schlechtesten und im mittleren Fall führt zu einer Bewertung der Eignung der Kurven für diese Aufgabe.

In Kapitel 2 wird die diskrete Form raumfüllender Kurven definiert, die in dieser Arbeit untersucht wird. Zudem werden dort die untersuchten zwei- und dreidimensionalen Kurven beschrieben und die wichtigsten bekannten Ergebnisse zu ihren Lokalitätseigenschaften vorgestellt. Der Einsatz der Kurven bzgl. der Partitionierung von Finite-Elemente-Netzen wird in Kapitel 3 betrachtet. Zunächst wird die Partitionierungsqualität in zweidimensionalen Gittern analysiert. Desweiteren wird die Eignung der Kurven für die Partitionierung unstrukturierter Netze an einem Satz von Benchmark-Netzen untersucht. In Kapitel 4 wird ein Algorithmus zur globalen Kontaktsuche auf Basis raumfüllender Kurven eingeführt und sein algorithmischer Aufwand analysiert. Zudem wird für diese Aufgabenstellung ein neues Qualitätsmaß für raumfüllende Kurven entwickelt. Für dieses werden Schranken zu verschiedenen Kurven erarbeitet. Die Integration der entwickelten Methoden in eine Umgebung zur expliziten Simulation nach der Finite-Elemente-Methode wird in Kapitel 5 beschrieben. Neben der Umsetzung der Algorithmen innerhalb der gegebenen industriellen Applikation werden die erzielten Ergebnisse der Parallelisierung und der effizienten Kontaktsuche dargestellt und diskutiert.

Kapitel 2

Diskrete raumfüllende Kurven

Stetige surjektive Abbildungen des Intervalls $[0,1]$ in den r-dimensionalen Würfel $[0,1]^r$ werden als raumfüllende Kurve bezeichnet. Zunächst wurde der Wertebereich des Einheitsquadrats betrachtet. Die erste stetige Abbildung einer Linie in ein Flächenstück ist von Peano 1890 definiert worden [63]. Die Konstruktion dieser Kurve ist in Abbildung 2.1 skizziert. Ein quadratisches Teilgebiet der Ebene wird in 3×3 Quadrate zerlegt, über die eine schlangenförmige Ordnung gelegt wird. Für die rekursive Verfeinerung wird in jedem der neun Quadrate wiederum eine schlangenförmige Ordnung gewählt. In jedem zweiten Quadrat wird die Kurve, welche die Ordnung repräsentiert, horizontal gespiegelt, so dass sich für jede beliebige Rekursionstiefe eine kontinuierliche Kurve ergibt. Einen historischen Überblick über die Entwicklung raumfüllender Kurven und deren mathematischer Eigenschaften sind in [69] zusammengefasst.

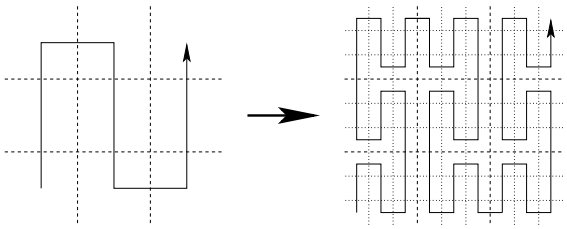


Abbildung 2.1: Geometrische Definition der Peano-Kurve

2.1 Definition der Kurven

Innerhalb dieser Arbeit betrachten wir Abbildungen zwischen diskreten Räumen die eine Approximation einer raumfüllenden Kurve beschreiben. Seien $x = (x_1, \ldots, x_r)$ und $y = (y_1, \ldots, y_r)$ Positionen im r-dimensionalen Raum und $||x - y||_\infty = \max\{|x_i - y_i| \;\; | i \in \{1, \ldots, r\}\}$ deren Abstand.

Definition 2.1 (Diskrete raumfüllende Kurven) *Sei curve* $: [0,1] \rightarrow [0,1]^r$ *eine raumfüllende Kurve.* $curve_n : \{1, \ldots, n^r\} \rightarrow \{1, \ldots, n\}^r$ *bezeichnen wir als* diskrete

raumfüllende Kurve, falls

(a) $curve_n$ *bijektiv ist* und

(b) $\left\| curve\left(\frac{k}{n^r}\right) - \frac{curve_n(k)}{n} \right\|_\infty \leq \frac{1}{n}$, *für alle* $k \in \{1, \ldots, n^r\}$.

Verbindet man die Bildpunkte gemäß ihrer Ordnung im Definitionsbereich ergibt sich eine Kurve durch die n^r Zellen des r-dimensionalen Gitters der Größe n. In dieser Arbeit beziehen sich alle Resultate auf diskrete raumfüllende Kurven, so dass im Folgenden auf die Spezifizierung *diskret* verzichtet wird. Aus der graphischen Darstellung ergibt sich auch der Begriff der *Indizierung* (oder Indizierungsschema), da über die Umkehrfunktion $curve^{-1}$ jeder dieser Gitterzellen ein Index zwischen 1 und n^r zugewiesen wird.

Die graphische Definition der Kurven erfolgt über die rekursive Aufteilung der Ebene über Geraden (bzw. Flächen im dreidimensionalen Raum), die im Folgenden als Separatoren bezeichnet werden. Teilgebiete, die durch die Separatoren begrenzt sind, werden als Zellen bezeichnet. Die Ordnung der Zellen wird durch eine Kurve skizziert. Sie beginnt in den folgenden Definitionen i. A. an der unteren linken Zelle.

Als *rekursive* raumfüllende Kurven werden solche bezeichnet, die über die rekursive Verfeinerung in jeweils vier quadratische Teilgebiete definiert sind [6]. Damit ergeben sich in der Konstruktion lediglich vertikale und horizontale Separatoren. Den Separatoren rekursiver raumfüllender Kurven werden Level zugeordnet, die invers zu ihrer Einfügung nummeriert werden. Damit gibt es im Gitter mit $(2^k)^2$ Zellen insgesamt $2 \cdot 2^k - 2$ Separatoren, von denen eine Hälfte vertikal und die andere Hälfte horizontal angeordnet ist. Von den vertikalen und horizontalen Separatoren sind jeweils 2^{k-1} vom Level 0 und genau einer, der mittlere, ist vom Level $k-1$. Das Level einer Kurve ist durch den höchsten Level der enthaltenen Separatoren gegeben, d. h. eine Kurve im $2^k \times 2^k$ Gitter ist vom Level $k-1$. Sind benachbarte Indizes jeweils benachbarten Zellen im Gitter zugewiesen, sprechen wir von einer kontinuierlichen Kurve.

2.1.1 Hilbert-Kurve

Die Hilbert-Kurve ist die am häufigsten genutzte und am meisten untersuchte raumfüllende Kurve. Sie wurde 1891 von Hilbert eingeführt [38] und gehört zu den rekursiven raumfüllenden Kurven. Die geometrische Definition der Kurve ist in Abbildung 2.2 angegeben. Die U-förmige Basisstruktur im 2×2-Gitter wird durch vier solcher Strukturen im 4×4-Gitter ersetzt. Dabei wird die Basisstruktur ggf. so rotiert oder gespiegelt, dass die Kurve einen Hamilton-Pfad durch die Zellen des Gitters bildet.

In Abbildung 2.3 ist die Hilbert-Kurve im 32×32-Gitter dargestellt. Aus Gründen der Übersichtlichkeit sind die Separatoren vom Level 0 und 1 ausgelassen. Die Ordnung der Zellen über den Verlauf der Kurve ist in einem Farbverlauf von rot zu blau dargestellt, um die Lokalitätseigenschaften hervorzuheben. So ist z. B. die rekursive Aufteilung in jeweils vier quadratische Teilgebiete gut sichtbar. Für die Abbildung, die durch

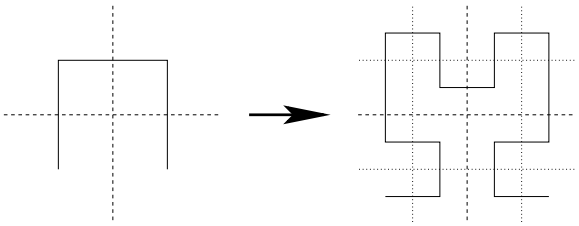


Abbildung 2.2: Geometrische Definition der Hilbert-Kurve

die Kurve definiert ist, gilt, dass von im Urbild-Raum benachbarten Werten auch die zugehörigen Bildwerte im Bild-Raum benachbart sind. Die Umkehrung gilt i. A. nicht, es ist jedoch in vielen Fällen so, dass die Nachbarn im Bild-Raum auch häufig im Urbild-Raum nahe zusammen liegen. Diese Nähe findet sich an Separatoren von niedrigem Level, während an hohen Leveln teilweise eine sehr große Distanz zu finden ist. So liegen die Zellen entlang des vertikalen mittleren Separators vom Level 4 sehr weit auseinander, was an der roten bzw. blauen Färbung der Kurve im unteren Bereich der Darstellung zu erkennen ist.

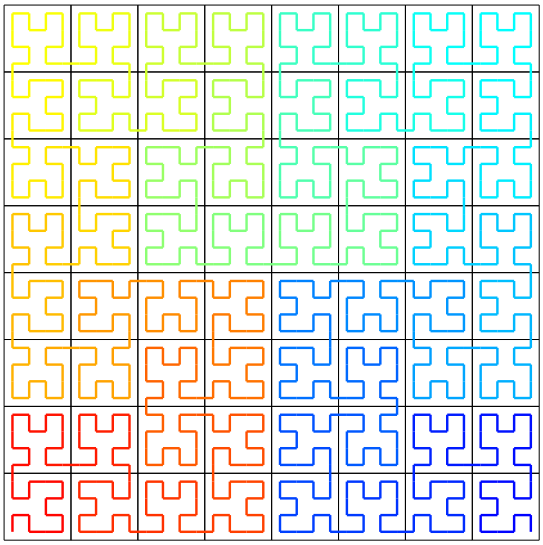


Abbildung 2.3: Hilbert-Kurve im 32×32-Gitter

Dreidimensionale Hilbert-Kurven Hilbert hat die nach ihm benannte Kurve nur für den zweidimensionalen Raum beschrieben. Sie stellt die einzige Möglichkeit dar, im zweidimensionalen Raum eine kontinuierliche rekursive raumfüllende Kurve über lediglich einer Basisstruktur zu definieren. Nimmt man diese Charakteritik als „Hilbert-Eigenschaft“, so ergeben sich 1536 Varianten der Hilbert-Kurve im dreidimensionalen Raum [3]. Für die Untersuchungen in dieser Arbeit wird die dreidimensionale Kurve betrachtet, deren geometrische Definition in Abbildung 2.4 angegeben ist.

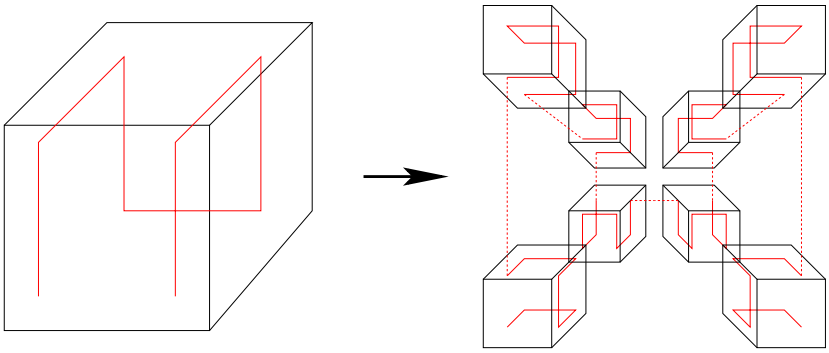


Abbildung 2.4: Untersuchte Variante der 3-dimensionalen Hilbert-Kurven

2.1.2 Lebesgue-Kurve

Die 1904 von Lebesgue definierte Abbildung bildet eine Kurve, die auf einer N-förmigen Basisstruktur aufbaut. Sie wird durch rekursive Vervielfältigung verfeinert, vergleichbar mit der Hilbert-Kurve. Im Gegensatz zur Hilbert-Kurve wird die Basisstruktur jedoch weder rotiert noch gespiegelt (vgl. Abb. 2.5). Auf der einen Seite ergibt sich dadurch eine Kurve mit deutlich mehr Symmetrie-Eigenschaften. Auf der anderen Seite sind die Bildwerte von im Urbild-Raum benachbarten Werten i. A. nicht benachbart. Es entstehen Sprünge innerhalb der Kurve, deren Länge von dem Level des Separators abhängt, an dem sie auftreten.

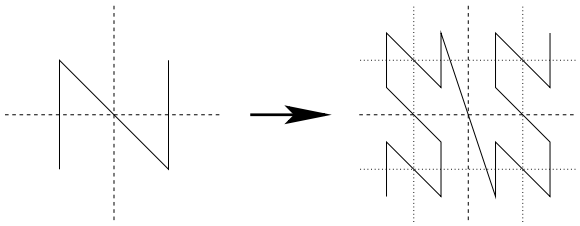


Abbildung 2.5: Geometrische Konstruktion der Lebesgue-Kurve

Abbildung 2.6 zeigt diesen Effekt in der Lebesgue-Kurve des 32×32-Gitters. Insbesondere an den mittleren Separatoren ergeben sich lange Sprünge. Für die Lebesgue-Kurve gilt, dass alle Zellen die weiter rechts oder weiter oben im Gitter liegen, einen höheren Index haben.

2.1.2.1 Besondere Eigenschaften

Wie in der Hilbert-Kurve haben viele benachbarte Zellen ähnliche Index-Werte bzgl. der Lebesgue-Kurve. Die Differenz der Index-Werte zweier benachbarter Zellen hängt in der Lebesgue-Kurve nur von dem Level und der Orientierung des trennenden Separators ab. Sie ergibt sich aus dem folgenden Lemma:

Lemma 2.1 *Seien x und y zwei benachbarte Zellen im Gitter, die durch einen Separator vom Level l getrennt sind.*

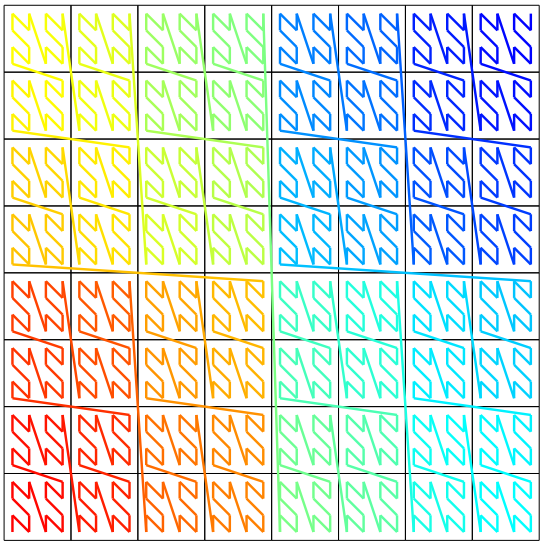


Abbildung 2.6: Lebesgue-Kurve im 32×32-Gitter

Für die Differenz ihrer Indizes, $\delta(x,y) = |\text{Lebesgue}^{-1}(x) - \text{Lebesgue}^{-1}(y)|$*, gilt*

$$\begin{aligned} \delta(x,y) = \quad U_l \quad &= \frac{2 \cdot 4^l + 1}{3} \qquad \textit{an einem horizontalen Separator und} \\ \delta(x,y) = \quad R_l \quad &= \frac{4 \cdot 4^l + 2}{3} \qquad \textit{an einem vertikalen Separator .} \end{aligned}$$

Beweis: Der Beweis erfolgt über eine vollständige Induktion über den Level l. Im Folgenden gelte $\text{Lebesgue}^{-1}(x) < \text{Lebesgue}^{-1}(y)$.

Induktionsanfang ($l = 0$):

Für Separatoren vom Level 0 gilt:

$$U_0 = 1$$
$$R_0 = 2$$

Dieser Abstand gilt für *alle* Zellenpaare an einem solchen Separator.

Induktionsschritt ($l \to l+1$):

Der Induktionsschritt erfolgt in zwei Teilen, für R_{l+1} und für U_{l+1}. Der Beweis für U_{l+1} erfolgt über alle R_i mit $i \leq l$ und der Beweis für R_{l+1} über alle U_i mit $i \leq l+1$.

$R_i (i \leq l) \to U_{l+1}$:

Es gelte $R_i = \frac{4 \cdot 4^i + 2}{3}$ für alle Zellenpaare an Separatoren vom Level $i \leq l$. Für den Induktionsschritt betrachten wir den Verlauf der Kurve: Seien x und y am linken Ende des horizontalen Separators vom Level $l+1$ gelegen. Der Pfad der Kurve durchläuft 2^{l+1} Zellen rechts von x und springt dann zur Zelle y. Abbildung 2.7 zeigt den Pfad, der über z verläuft, der Zelle mit dem höchsten Index im ersten Quadranten. y ist die Zelle mit dem kleinsten Index im zweiten Quadranten. Die Häufigkeit eines vertikalen Separators

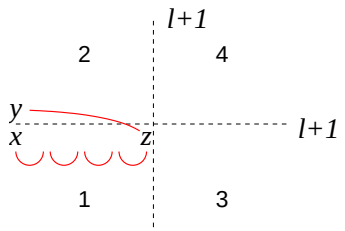


Abbildung 2.7: Pfad zwischen benachbarten Zellen x und y am horizontalen Separator vom Level l

vom Level $i \leq l$ auf diesem Weg ist 2^{l-i}. Damit kann die Distanz der Lebesgue-Indizes zwischen den Zellen x und y aus den Einzeldistanzen an den vertikalen Separatoren bestimmt werden plus einer Zelle für den Sprung von der Zelle z auf die Zelle y.

$$\begin{aligned}
U_{l+1} &= \sum_{i=0}^{l} 2^{l-i} \cdot R_i + 1 \\
&= \sum_{i=0}^{l} 2^{l-i} \cdot \left(\frac{4}{3} \cdot 4^i + \frac{2}{3}\right) + 1 \\
&= \sum_{i=0}^{l} 2^{l-i} \cdot \frac{4}{3} \cdot 4^i + \sum_{i=0}^{l} 2^{l-i} \cdot \frac{2}{3} + 1 \\
&= \sum_{i=0}^{l} 2^{l} \cdot \frac{4}{3} \cdot 2^i + \sum_{i=0}^{l} 2^{i} \cdot \frac{2}{3} + 1 \\
&= 2^l \cdot \frac{4}{3} \cdot \left(2^{l+1} - 1\right) + \frac{2}{3} \cdot \left(2^{l+1} - 1\right) + \frac{3}{3} \\
&= \frac{2 \cdot 4^{l+1} + 1}{3}
\end{aligned}$$

Liegen die Zellen x und y nicht am linken Ende des Separators, bleibt die Anzahl der übersprungenen Separatoren eines jeden Levels $i \leq l$ gleich. Es ändert sich lediglich die Ordnung, in der die Separatoren durchlaufen werden.

$U_i (i \leq l+1) \rightarrow R_{l+1}$:

Der Induktionsschritt für die Distanz an vertikalen Separatoren vom Level $l+1$ erfolgt analog zu dem oben gezeigten. In diesem Fall werden jedoch jeweils 2^{l-i+1} horizontale Separatoren von jedem Level $i \leq l+1$ durchlaufen. Es gelte $U_i = \frac{2 \cdot 4^i + 1}{3}$ für alle $i \leq l+1$. Dann folgt:

$$\begin{aligned}
R_{l+1} &= \sum_{i=0}^{l+1} 2^{l-i+1} \cdot U_i + 1 \\
&= \sum_{i=0}^{l+1} 2^{l-i+1} \cdot \left(\frac{2}{3} \cdot 4^i + \frac{2}{3}\right) + 1 \\
&= \sum_{i=0}^{l+1} 2^{l-i+1} \cdot \frac{2}{3} \cdot 4^i + \sum_{i=0}^{l+1} 2^{l-i+1} \cdot \frac{1}{3} + 1
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{l+1} 2^{l+1} \cdot \frac{2}{3} \cdot 2^i + \sum_{i=0}^{l+1} 2^i \cdot \frac{1}{3} + 1 \\
&= 2^{l+1} \cdot \frac{2}{3} \cdot \left(2^{l+2} - 1\right) + \frac{1}{3} \cdot \left(2^{l+2} - 1\right) + 1 \\
&= \frac{4 \cdot 4^{l+1} + 2}{3}
\end{aligned}
$$

Wiederum ist die Anzahl der übersprungenen horizontalen Separatoren vom Level $i \leq l + 1$ unabhängig von der relativen Position der benachbarten Zellen am betrachteten vertikalen Separator. □

Graphisch lässt sich die Distanz über die durchlaufenen quadratischen Gebiete beschreiben, die bei der rekursiven Konstruktion der Kurve entstanden sind. Im Falle des vertikalen Separators vom Level l wird für alle $i \leq l$ jeweils ein Quadrat der Größe $2^i \times 2^i$ durchlaufen. Damit gilt:

$$
\begin{aligned}
R_l &= \sum_{i=0}^{l} 4^i + 1 \\
&= \frac{4^{l+1} - 1}{3} + 1 \\
&= \frac{4 \cdot 4^l + 2}{3}
\end{aligned}
$$

Beim Weg zu einer benachbarten Zelle, die durch einen horizontalen Separator vom Level l getrennt wird, werden für alle $i \leq l$ jeweils Rechtecke der Größe $2^i \times 2^{i-1}$ durchlaufen.

Dreidimensionale Lebesgue-Kurve Im Gegensatz zu der Hilbert-Kurve ist die Erweiterung der Lebesgue-Kurve für den drei- und höherdimensionalen Raum eindeutig. Für jede weitere Raumdimension wird die Basisstruktur verdoppelt und das Ende der ersten mit dem Beginn der zweiten verbunden. Abbildung 2.8 zeigt die Struktur der dreidimensionalen Kurve. Für eine Verfeinerung wird in allen acht Teilwürfeln wiederum die gleiche Struktur eingesetzt.

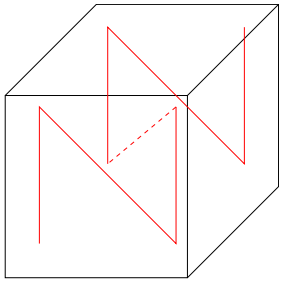


Abbildung 2.8: Dreidimensionale Erweiterung der Lebesgue-Kurve

2.1.3 Sierpiński-Kurve

Im Jahre 1912 hat Sierpiński eine weitere raumfüllende Kurve vorgestellt. Ihre geometrische Konstruktion ist in Abbildung 2.9 angegeben. Die Aufteilung des zweidimensionalen Raumes erfolgt über achsenparallele *und* diagonale Separationslinien. Somit gehört diese Kurve nicht zu den rekursiven raumfüllenden Kurven. Jedes der vier recht-

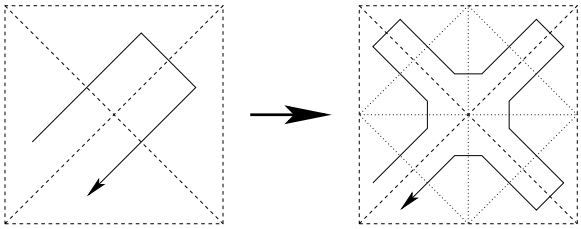


Abbildung 2.9: Geometrische Definition der Sierpiński-Kurve

winkligen, gleichschenkligen Dreiecke wird durch vier kleinere Dreiecke ersetzt und in diesen eine kontinuierliche Ordnung definiert. Wie die Hilbert-Kurve ergibt sich damit ein kontinuierlicher Verlauf der Kurve im Raum. Im Gegensatz zu den beiden anderen ist sie jedoch zyklisch und bildet damit einen Hamilton-Kreis über die Flächen des triangulierten Gebietes. Die Sierpiński-Kurve nach der vierten Verfeinerung ist in Abbildung 2.10 dargestellt. Wiederum ist lediglich ein grobes 8×8-Gitter zur besseren Orientierung eingezeichnet.

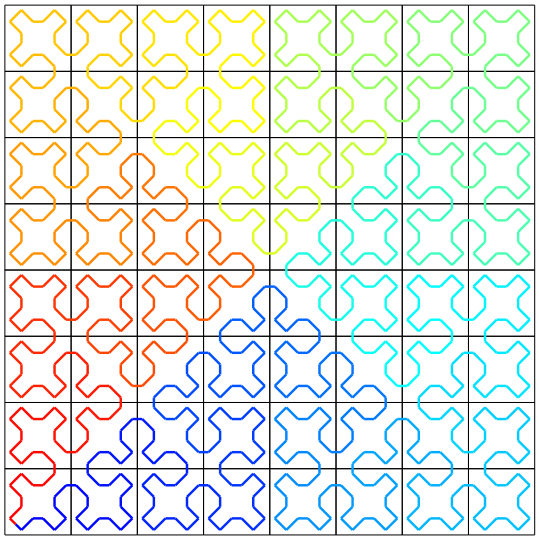


Abbildung 2.10: Sierpiński-Kurve nach vier Verfeinerungen

Bei der Sierpiński-Kurve handelt sich um keine *diskrete* raumfüllende Kurve, da sie keine Abbildung in ein Gitter darstellt. Die Aufteilung des Raumes ist jedoch geeignet, um eine Ordnung von Punkten im Raum zu definieren, die für die in Kapitel 3 beschriebene Partitionierung von unstrukturierten FE-Netzen eingesetzt werden kann. Zudem ist es möglich, über die Struktur der Sierpiński-Kurve eine Indizierung im Gitter zu

definieren, wie die H-Indizierung.

2.1.3.1 H-Indizierung

Niedermeier et al. haben 1997 die H-Indizierung vorgestellt. Ihre ursprüngliche Definition erfolgt über Pfade in Gittern bzw. in Teilgebieten aus Gittern [55]. Aufgrund der großen Verwandtschaft zur Struktur der Sierpiński-Kurve, kann die geometrische Konstruktion auch aus dieser Kurve erfolgen. Nach der rekursiven Verfeinerung bis zum gewünschten Level werden alle Strukturen aus 16 Dreiecken in 16 Quadrate aufgeteilt und die Kurve gemäß der in Abbildung 2.11 dargestellten Transformation angepasst. Die resultierende Kurve im 32×32-Gitter ist in Abbildung 2.12 dargestellt. Wie in der

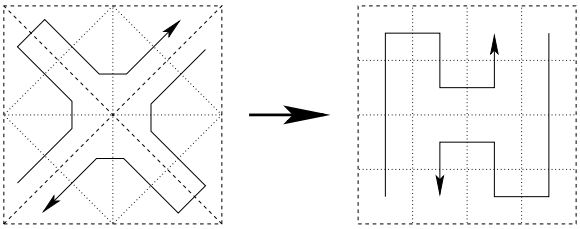


Abbildung 2.11: Ableitung der H-Indizierung aus der Sierpiński-Kurve

Sierpiński-Kurve ergibt sich ein Hamilton-Kreis, in diesem Fall als Indizierung über den Zellen des Gitters. Die diagonalen Separationslinien innerhalb der Konstruktion sind durch den farblichen Verlauf der Kurve gut sichtbar.

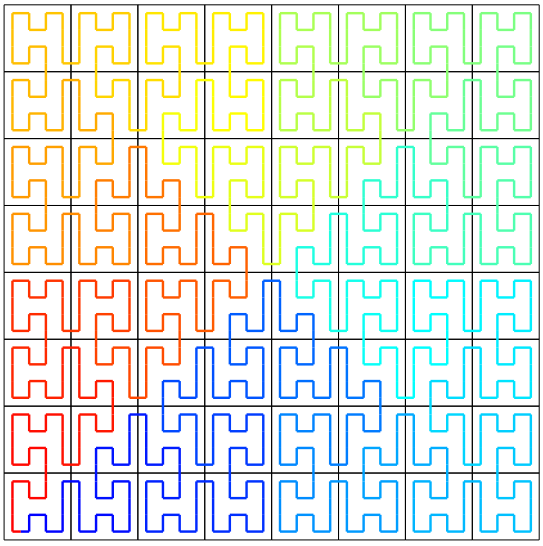


Abbildung 2.12: H-Indizierung im 32×32-Gitter

Auch wenn die H-Indizierung eine Abbildung der geforderten Form ist, ist sie aufgrund der diagonalen Separationen keine *rekursive* raumfüllende Kurve. Eine zyklische rekursive raumfüllende Kurve ist im folgenden Abschnitt definiert.

2.1.4 $\beta\Omega$-Indizierung

Die $\beta\Omega$-Indizierung ist eine neue raumfüllende Kurve. Sie ist mit dem Ziel entwickelt worden, bessere Lokalitätseigenschaften im Hinblick der in dieser Arbeit untersuchten Anwendungsgebiete zu erzielen [81]. Sie nutzt die gleiche U-förmige Basisstruktur wie die Hilbert-Kurve. Es existieren jedoch zwei Subtypen, die zu unterschiedlichen Verfeinerungen führen. Die beiden Subtypen und ihre Verfeinerungsregeln sind in Abbildung 2.13 dargestellt. Die roten Punkte markieren die Richtung, in die der Übergang zum benachbarten Teilgitter zu erfolgen hat und charakterisieren damit den Subtyp. Der obere Subtyp wird als *gebogene*, der untere als *gerade* Basisstruktur bezeichnet.

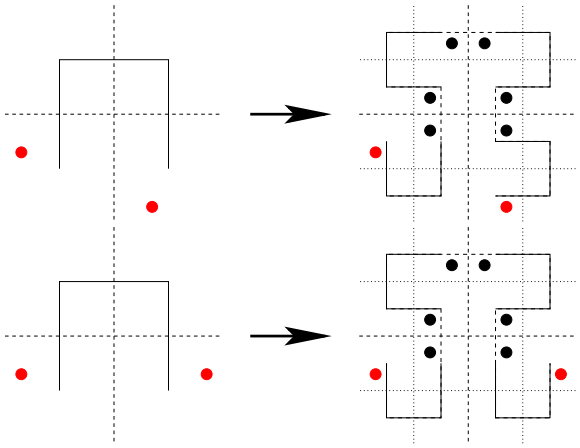


Abbildung 2.13: Geometrische Definition der $\beta\Omega$-Indizierung

Die beiden Endpunkte der gebogenen Struktur liegen an benachbarten Kanten des umschließenden Quadrates, die Endpunkte der geraden Struktur an gegenüberliegenden. Die Verfeinerung der gebogenen Struktur führt zu drei gebogenen und einer geraden Basisstruktur, die der geraden Struktur zu vier gebogenen Substrukturen. Damit die Kurve einen Hamilton-Kreis über die Zellen des Gitters definiert, ist eine spezielle Ordnung in der ersten Verfeinerung notwendig (vgl. Abb. 2.14). Sie beinhaltet ebenfalls vier gebogene Basisstrukturen. Diese Ordnung ist mit der in Moores Version der

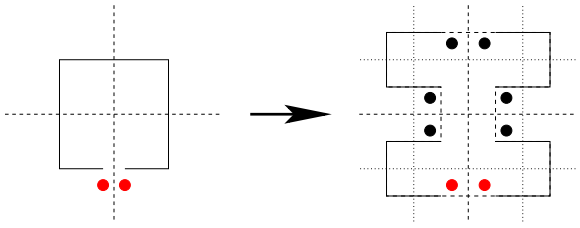


Abbildung 2.14: Ordnung in der ersten Verfeinerung der $\beta\Omega$-Indizierung

Hilbert-Kurve identisch. Die Verfeinerung in Moores Kurve erfolgt jedoch über die Regel der Hilbert-Kurve [69]. In Abbildung 2.15 ist die $\beta\Omega$-Indizierung im 32×32-Gitter dargestellt.

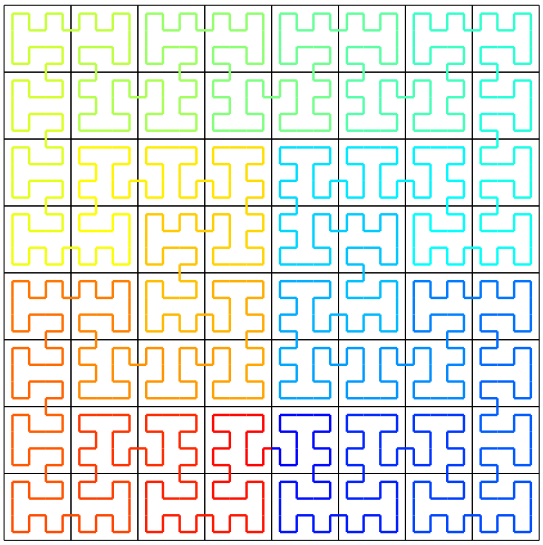


Abbildung 2.15: $\beta\Omega$-Indizierung im 32×32-Gitter

Der Name der Kurve ergibt sich aus dem Pfad innerhalb der beiden Basisstrukturen. Nach einer Glättung ergeben sich Formen, die den griechischen Buchstaben β und Ω ähneln (vgl. Abb. 2.16).

Abbildung 2.16: Bezeichnung der Kurve

2.1.4.1 Besondere Eigenschaften

Bei allen anderen bisher vorgestellten Kurven liegen die Übergänge von einem Quadranten in den benachbarten an den relativen Positionen 0 bzw. 1 der Separatoren, d. h. an zwei der Ecken des umschließenden Quadrats. Bei der Sierpiński- und H-Kurve gilt diese Beobachtung auch an den diagonalen Separatoren. Bei der $\beta\Omega$-Kurve konvergiert die relative Position eines Übergangs gegen 1/3 bzw. 2/3.

Lemma 2.2 *Sei eine Teilkurve der $\beta\Omega$-Kurve vom Level l gegeben. Für die relative Position α des Übergangs zu einem benachbarten Quadranten von gleichem Level gilt*

$$\alpha(l) = \frac{2^{l+2}-1}{3} \cdot \frac{1}{2^{l+2}} \quad \textit{oder} \quad \alpha(l) = 1 - \frac{2^{l+2}-1}{3} \cdot \frac{1}{2^{l+2}} \ .$$

Beweis: Wir betrachten lediglich die Positionen auf der ersten Hälfte einer Kante, also $\alpha < 1/2$. Der Beweis für $\alpha > 1/2$ erfolgt analog. Die Position des Übergangs liegt

immer in der Mitte der Zelle, in der die Teilkurve vom Level l endet. Damit ergibt sich aus der Definition der geraden und der gebogenen Basisstruktur eine relative Position von (vgl. Abb. 2.13):

$$\alpha(0) = 1/4$$

Für beide Verfeinerungsregeln gilt, dass der Endpunkt alternierend im oberen und unteren Quadranten bzgl. der nächstkleineren Level liegt, bzw. rechts und links bei Übergängen an einem horizontalen Separator. Da sich die Kantenlänge bei jeder Verfeinerung halbiert, gilt für den Übergang aus einer Teilkurve vom Level l:

$$\alpha(l) = \sum_{i=0}^{l} \frac{-1^i}{2^{i+2}} = \frac{1}{4} - \frac{1}{8} + \frac{1}{16} + \cdots + \frac{-1^l}{2^l + 2} = \frac{2^{l+2} - 1}{3} \cdot \frac{1}{2^{l+2}}$$

□

Da sich die beiden Subtypen bzgl. der Lokalitäteigenschaften ihrer Teilkurven unterscheiden, ist die Häufigkeit der Substrukturen innerhalb einer Kurve von höherem Level wichtig. Das folgende Lemma zeigt, dass das Verhältnis von gebogenen zu geraden Strukturen in Kurven von hohem Level gegen 4 : 1 konvergiert.

Lemma 2.3 *Seien $l_1, l_2 \in \mathbb{N}_0$ mit $l_1 < l_2$ und $l = l_2 - l_1$. Die Anzahl der geraden Strukturen $\mathcal{N}_{\text{ger}}$ vom Level l_1 innerhalb der $\beta\Omega$-Kurve vom Level l_2 beträgt*

$$\begin{aligned} \mathcal{N}_{\text{ger}}(l) &= \frac{4^l + 4}{5} \quad \textit{, falls } l \textit{ gerade und} \\ \mathcal{N}_{\text{ger}}(l) &= \frac{4^l - 4}{5} \quad \textit{, falls } l \textit{ ungerade .} \end{aligned}$$

Für die Anzahl der gebogenen Strukturen folgt

$$\begin{aligned} \mathcal{N}_{\text{geb}}(l) &= 4 \cdot \frac{4^l - 1}{5} \quad \textit{, falls } l \textit{ gerade und} \\ \mathcal{N}_{\text{geb}}(l) &= 4 \cdot \frac{4^l + 1}{5} \quad \textit{, falls } l \textit{ ungerade .} \end{aligned}$$

Beweis: Der Beweis erfolgt über vollständige Induktion über den Level l. Da sich die Anzahl der Strukturen höheren Levels bei einer weiteren Verfeinerung nicht mehr ändert, ist die absolute Größe von l_1 und l_2 unerheblich.

Induktionsanfang ($l \in \{1, 2\}$):

Für die Level 1 und 2 ergeben sich die Häufigkeiten

$$\begin{aligned} \mathcal{N}_{\text{ger}}(1) = 0 \quad &\text{und} \quad \mathcal{N}_{\text{geb}}(1) = 4 \quad \text{und} \\ \mathcal{N}_{\text{ger}}(2) = 4 \quad &\text{und} \quad \mathcal{N}_{\text{geb}}(2) = 12 \ . \end{aligned}$$

Induktionsschritt ($l \to l + 1$):

Bei der Verfeinerung einer geraden Basisstruktur entstehen vier gebogene Strukturen

von kleinerem Level. Aus einer gebogenen Struktur entstehen drei gebogene und eine gerade. Daher gilt immer

$$\begin{aligned} \mathcal{N}_{\text{ger}}(l+1) &= \mathcal{N}_{\text{geb}}(l) \quad \text{und} \\ \mathcal{N}_{\text{geb}}(l+1) &= 4 \cdot \mathcal{N}_{\text{ger}}(l) + 3 \cdot \mathcal{N}_{\text{geb}}(l) \ . \end{aligned}$$

Für l gerade folgt

$$\begin{aligned} \mathcal{N}_{\text{ger}}(l+1) &= 4 \cdot \frac{4^l - 1}{5} \\ &= \frac{4^{l+1} - 1}{5} \end{aligned}$$

und

$$\begin{aligned} \mathcal{N}_{\text{geb}}(l+1) &= 4 \cdot \frac{4^l + 4}{5} + 3 \cdot 4 \cdot \frac{4^l - 1}{5} \\ &= 4 \cdot \frac{(1+3) \cdot 4^l + 4 - 3}{5} \\ &= 4 \cdot \frac{4^{l+1} + 1}{5} \ . \end{aligned}$$

Der Induktionsschritt für den Fall, dass l ungerade ist, erfolgt analog. □

2.2 Eigenschaften raumfüllender Kurven

Raumfüllende Kurven werden in vielen Gebieten der Informatik eingesetzt. Dabei wird je nach Anwendung entweder die Abbildung selbst oder ihre Umkehrfunktion, d. h. die Indizierung des Raumes, genutzt. Der erste Fall ist die Einbettung eines eindimensionalen Feldes in den r-dimensionalen Raum, der zweite eine Einbettung in umgekehrter Richtung. Nach der Beschreibung einiger Anwendungsgebiete sind die wichtigsten Ergebnisse zu den Eigenschaften raumfüllender Kurven angegeben. Zudem sind die Beweisansätze skizziert, um die gängigen Methoden der Beweisführung in diesem Bereich aufzuzeigen.

Die Abbildung in den Raum wird u. a. für die Entwicklung paralleler Algorithmen auf Parallelrechnern mit gitterartiger Netzwerkstruktur eingesetzt. Beispiele sind Sortieralgorithmen, bei denen ein eindimensionales Feld gegeben ist und benachbarte Partitionen im Feld über kurze Kommunikationspfade im Gitter verbunden sein sollen [33, 71]. Ein weiteres untersuchtes Einsatzgebiet ist die Einbettung von Suchbäumen (wie beim *branch-and-bound* oder *backtracking*) in das Gitter eines Parallelrechners. Dabei sollen Sohnknoten möglichst nahe zu ihrem Vater platziert sein. Eine Analyse für die Laufzeit einer Baumsuche bei der Einbettung über die Hilbert-Kurve ist in [43] gegeben. Aus diesen Aufgabenstellungen ist die Definition der *Lokalität* raumfüllender Kurven entstanden.

Die Indizierung des Raumes über Kurven wird dann eingesetzt, wenn eine eindimensionale Struktur gefordert ist, in der räumliche Nachbarschaften weitestgehend erhalten

bleiben. Die wichtigste Anwendung in diesem Bereich ist die Speicherung mehrdimensionaler Datenbanken auf Hintergrundspeichern [6, 42, 53]. Anfragen in solchen Datenbanken werden typischerweise durch Intervalle über mehrere Eigenschaften beschrieben, die als Polytope im r-dimensionalen Raum angesehen werden können. Für die Beantwortung müssen damit alle Datenblöcke aus dem Hintergrundspeicher gelesen werden, die in diesem Polytop enthalten sind. Da die Zugriffszeit auf Daten im Hintergrundspeicher von den Zeiten für die Positionierung des Lesekopfes dominiert wird, stellt sich die Frage, welche Anordnung der Daten zu den wenigsten Positionierungen des Lesekopfes führt? Diese Anforderung wird über die Anzahl der *Zusammenhangskomponenten* einer Kurve innerhalb eines Polytopes untersucht.

Ein weiteres Anwendungsgebiet für die Indizierungseigenschaften der raumfüllenden Kurven ist die Partitionierung von strukturierten und unstrukturierten Netzen für parallele Berechnungen. Über die raumfüllende Kurve wird eine Indizierung der Knoten des Netzes durchgeführt. Partitionen werden über die Aufteilung der Indizes in Intervalle gebildet. Damit entsprechen die Partitionen disjunkten Teilpfaden der raumfüllenden Kurve. In mehreren Arbeiten wurde eine zufriedenstellende Qualität der resultierenden Partitionierung festgestellt [4, 31, 62, 64]. Eine weitergehende Analyse von Partitionierungen auf Basis raumfüllender Kurven wird in Kapitel 3 durchgeführt. Dort wird auch ein entsprechendes Maß für die Qualität von Partitionierungen definiert.

2.2.1 Lokalität

Die Lokalität von raumfüllenden Kurven ist über den geometrischen Abstand der Punkte auf der Kurve definiert. Das Maß ist von Gotsman und Lindenbaum für die euklidische Distanz eingeführt worden [28]. An dieser Stelle nutzen wir eine Verallgemeinerung, die auch die Maximum- und Manhattan-Metrik beinhaltet [2].

Sei $x = (x_1, \ldots, x_r)$ die Position im r-dimensionalen Raum. Für $p \geq 1$ sei $||x||_p$ die p-Norm, d. h. $||x||_p = (\sum_{k=1}^{r} |x_k|^p)^{1/p}$ und $||x||_\infty = \max\{|x_1|, \ldots, |x_r|\}$. Daraus lassen sich die Distanzfunktionen $d_p(x, y) = ||x - y||_p$ für $x, y \in \mathbb{R}^r$ ableiten. Die bisher veröffentlichten Ergebnisse beziehen sich auf die Sonderfälle d_1 (Manhattan-Metrik), d_2 (euklidische Distanz) und d_∞ (Maximum-Metrik).

Definition 2.2 (Lokalität) *Sei $curve_n : \{1, \ldots, n^r\} \to \{1, \ldots, n\}^r$ eine bijektive Abbildung in den r-dimensionalen Raum. Die Lokalität dieser Kurve ist definiert als*

$$\mathcal{L}_p^{curve_n} = \max_{i,j \in \{1,\ldots,n^r\}, i \neq j} \frac{d_p(curve_n(i), curve_n(j))^r}{|i - j|} \ . \tag{2.1}$$

Von besonderem Interesse ist die Lokalität im Grenzfall von $n \to \infty$. Die Verallgemeinerung der Metrik auf eine Familie von Kurven $curve = \{curve_n | n \in \mathbb{N}\}$ ist

$$\mathcal{L}_p^{curve} = \lim_{n \to \infty} \mathcal{L}_p^{curve_n} \ . \tag{2.2}$$

Viele der unten dargestellten Schranken für allgemeine oder spezielle Kurven sind direkt über die geometrischen Distanzen angegeben und sind somit exakter, da sie additive

Konstanten berücksichtigen. An dieser Stelle wird jedoch lediglich das in der Definition beschriebene relative Verhältnis aus geometrischer Entfernung und Index-Distanz dargestellt, damit sich die Ergebnisse direkt vergleichen lassen.

2.2.1.1 Untere Schranken

Gotsman und Lindenbaum haben gezeigt, dass für jede kontinuierliche raumfüllende Kurve $curve_n$ im r-dimensionalen Raum

$$\mathcal{L}_2^{curve_n} > (2^r - 1)\left(1 - \frac{1}{n}\right)^r \tag{2.3}$$

eine untere Schranke für die Lokalität ist [28]. Der Beweis erfolgt über die Betrachtung der Eckpunkte des r-dimensionalen Würfels und deren Indizes. Für den zweidimensionalen Fall ergibt sich damit eine Schranke von $\mathcal{L}_2^{curve} > 3$ für beliebig große Kurven. Dieser Wert kann mit einer analogen Betrachtung auf $\mathcal{L}_2^{curve} > 3.25$ erhöht werden, wenn neun statt vier Punkte innerhalb des Quadrates betrachtet werden [28].

Niedermeier et al. haben verschiedene untere Schranken für den zweidimensionalen Raum bewiesen [55]. Für kontinuierliche Kurven gilt für die euklidische und die Maximum-Metrik $\mathcal{L}_{2,\infty}^{curve} \geq 3.5$. Eine untere Schranke in der Manhattan-Metrik ist $\mathcal{L}_1^{curve} \geq 6.5$. In der gleichen Arbeit zeigen sie Schranken für zyklische Kurven von $\mathcal{L}_{2,\infty}^{curve} \geq 4$ und $\mathcal{L}_1^{curve} \geq 8$. Alle Beweise betrachten charakteristische Punkte (meist Eckpunkte) in der quadratischen Grundfläche. Für diese können jeweils Schranken für den geometrischen Abstand (d_1, d_2 bzw. d_∞) und die Index-Differenz angegeben werden.

2.2.1.2 Obere Schranken

Obere Schranken zur Lokalität von raumfüllenden Kurven wurden für die Hilbert-Kurve, die H-Indizierung und die $\beta\Omega$-Indizierung untersucht. Für r-dimensionale Hilbert-Kurven gilt eine obere Schranken von [28]

$$\mathcal{L}_2^{\text{Hilbert}} < 2^r(r+3)^{r/2} \ . \tag{2.4}$$

Für den Beweis wird für einen beliebigen Pfad der Länge m ein $k \in \mathbb{N}$ so gewählt, dass $(2^r)^{k-1} < m \leq (2^r)^k$ gilt. Aufgrund der Form der Hilbert-Kurve, sind die r-dimensionalen Würfel mit Kantenlänge 2^k, in denen der Pfad beginnt und endet, benachbart *und* der Pfad liegt vollständig innerhalb dieser beiden Würfel. Aus der minimalen Pfadlänge und dem maximalen Durchmesser ergibt sich die Schranke. Aufgrund der notwendigen Voraussetzungen, ist die so erzielte Schranke für alle kontinuierlichen rekursiven raumfüllenden Kurven gültig. Für den Sonderfall $r = 2$ ergibt sich ein Wert von 20. Dieser kann auf $\mathcal{L}_2^{\text{Hilbert}} < 6\frac{2}{3}$ reduziert werden, wenn anstelle von zwei Quadraten der Größe $2^k \times 2^k$ alle Anordnungen mit bis zu fünf Quadraten der Größe $2^{k-1} \times 2^{k-1}$ betrachtet werden, die in der Hilbert-Kurve auftreten können [28]. Eine weitere Verbesserung auf $\mathcal{L}_2^{\text{Hilbert}} \leq 6\frac{1}{2}$ kann erzielt werden, wenn die möglichen Teilpfade in der nächstfeineren Hilbert-Kurve betrachtet werden [3].

Ein induktiver Beweis ist möglich, wenn von der oberen Schranke in $curve_n$ auf die obere Schranke in $curve_{2n}$ geschlossen werden kann. Ein solcher Beweisansatz führt für die Manhattan-Metrik zu $\mathcal{L}_1^{\text{Hilbert}} \leq 9$ [15, 55]. Dazu seien i, j die Indizes in der Hilbert$_{2n}$-Kurve und i', j' die Positionen im gröberen Hilbert$_n$. Die Induktion erfolgt über die Schranke für den Abstand von $d_1(i, j) \leq 2 \cdot d_1(i', j') + 2$.

Innerhalb der zweidimensionalen Hilbert-Kurve kann ein Pfad der Länge m konstruiert werden, dessen euklidischer Durchmesser gegen $\sqrt{6m}$ konvergiert [28] und ein Pfad, dessen d_1-Distanz gegen $\sqrt{9m}$ konvergiert [56]. Damit gilt, dass

$$\begin{aligned} \mathcal{L}_2^{\text{Hilbert}} &\in \left[6, 6\frac{1}{2}\right] \quad \text{und} \\ \mathcal{L}_1^{\text{Hilbert}} &= 9 \ . \end{aligned}$$

Ebenfalls über einen induktiven Beweis können gute Schranken für die H-Indizierung gefunden werden. Es gilt $\mathcal{L}_{2,\infty}^{\text{H-Ind}} \leq 4$ und $\mathcal{L}_1^{\text{H-Ind}} \leq 8$. Aus diesem Ergebnis und den unteren Schranken für zyklische raumfüllende Kurven folgt, dass die H-Indizierung ein asymptotisch[1] optimaler Vertreter dieser Klasse an Kurven ist.

Eine Analyse zur $\beta\Omega$-Indizierung erfolgt analog zur Hilbert-Kurve. Für einen Pfad der Länge m wird $k \in \mathbb{N}$ so gewählt, dass $4^{k-1} < m \leq 4^k$ gilt. Es werden alle Anordnungen von Quadraten der Größe $2^{k-2} \times 2^{k-2}$ gemäß der $\beta\Omega$-Kurve untersucht. Es ergibt sich eine obere Schranke von $\mathcal{L}_2^{\beta\Omega-\text{Ind}} < 5\frac{2}{3}$. Zudem kann ein schlechter Fall konstruiert werden, der $\mathcal{L}_2^{\beta\Omega-\text{Ind}} \geq 5$ zeigt [81]. Es folgt

$$\mathcal{L}_2^{\beta\Omega-\text{Ind}} \in \left[5, 5\frac{2}{3}\right] \ .$$

Daraus ergibt sich eine Ordnung für die untersuchten Kurven von

$$\mathcal{L}_2^{\text{H-Ind}} < \mathcal{L}_2^{\beta\Omega-\text{Ind}} < \mathcal{L}_2^{\text{Hilbert}} \tag{2.5}$$

im zweidimensionalen Raum. Weitere analytische Schranken, insbesondere zu dreidimensionalen Kurven, finden sich in [28, 56].

2.2.2 Zusammenhangskomponenten

Die Anzahl an Zusammenhangskomponenten in einer Kurve (engl.: *clustering property*), die in einem gegebenen Polyeder enthalten sind, sind sowohl im mittleren als auch im schlechtesten Fall untersucht worden.

2.2.2.1 Mittlerer Fall

Die meisten Untersuchungen zu dieser Fragestellung beschäftigen sich mit der Struktur der Hilbert-Kurve. Dieser Abschnitt gibt die wichtigsten Ergebnisse wieder. Für $r \in \mathbb{N}$

[1] In [55, 56] sind die Distanzen in den drei Metriken direkt angegeben, so dass der Abstand zur unteren Schranke exakt gegeben ist.

sei $\mathcal{N}_r$ die zu erwartende Anzahl an Teilpfaden einer Kurve, die im Mittel in einem r-dimensionalen Polyeder verlaufen.

Jagadish hat gezeigt, dass im zweidimensionalen Raum die mittlere Anzahl an Teilpfaden für eine Anfrageregion der Größe $k \times k$ für $k \in [2, 3]$ von unten gegen k konvergiert, d. h. es gilt $\mathcal{N}_2 \leq k$ für solche quadratische Anfragen [42]. Für jeden Teilpfad gilt, dass er an einer Stelle in die Anfrageregion eintritt und an einer zweiten austritt. Damit ist die Anzahl der Teilpfade halb so groß wie die der Schnitte des Pfades mit den Rändern der Anfrageregion. Der Beweis erfolgt über die Untersuchung aller möglichen Positionen, an der eine quadratische Anfrageregion der Größe $k \times k$ liegen kann. Für jede Kante des Hilbert-Pfades wird untersucht, durch wie viele dieser Anfrageregionen sie geschnitten wird. Das Verhältnis aus der summierten Anzahl an Teilpfaden zu der Anzahl aller Anfrageregionen ergibt die mittlere Zahl an Zusammenhangskomponenten.

Moon et al. haben diese Aussage auf Quadrate mit Kantenlänge 2^k erweitert: Seien $k, n \in \mathbb{N}$. Für quadratische Anfrageregionen der Größe $2^k \times 2^k$ in einer Kurve vom Level $n + k - 1$ (d. h. über $2^{n+k} \times 2^{n+k}$ Zellen) sind im Mittel

$$\mathcal{N}_2 = \frac{(2^n - 1)^2 2^{3k} + (2^n - 1)2^{2k} + 2^n}{(2^{k+n} - 2^k + 1)^2} \tag{2.6}$$

Teilpfade enthalten [53]. Der Beweisansatz ist analog zu dem oben skizzierten, d. h. es werden alle möglichen Fälle aufgezählt. In der gleichen Arbeit haben die Autoren auch eine Abschätzung für den r-dimensionalen Raum gegeben. Sei S_q die Oberfläche einer polyedrischen Anfrageregion q mit achsenparallelen Flächen in der Hilbert-Kurve vom Level n. Für die Anzahl der Zusammenhangskomponenten innerhalb q gilt

$$\lim_{n \to \infty} \mathcal{N}_r = \frac{S_q}{2r} \ . \tag{2.7}$$

Damit teilt eine Anfrageregion die Hilbert-Kurve im Mittel in weniger Zusammenhangskomponenten als die Lebesgue-Kurve [53]. Der Beweis für diese Aussage erfolgt über die Orientierung der Pfadsegmente der Hilbert-Kurve im Raum. In großen Kurven ist die Segmenthäufigkeit in jeder Richtung gleich groß. Daraus kann auf die Wahrscheinlichkeit geschlossen werden, mit der ein Punkt am Rand der Anfrageregion einen Index-Nachbarn außerhalb der Region besitzt.

Für den Spezialfall des r-dimensionalen Würfels mit Kantenlänge s folgt, dass in diesem im Mittel

$$\lim_{n \to \infty} \mathcal{N}_r = s^{r-1} \tag{2.8}$$

Zusammenhangskomponenten der Hilbert-Kurve liegen.

2.2.2.2 Schlechtester Fall

Asano et al. haben untersucht, wie viele Zusammenhangskomponenten im schlechtesten Fall notwendig sind, wenn eine Vergrößerung der Region um einen konstanten Faktor erlaubt ist [6]. Ihre Arbeiten beziehen sich alle auf den zweidimensionalen Raum. Für eine im vorhinein bekannte Anfrageregion der Größe $k \times k$ konnten sie eine Kurve

angeben, die mit höchstens 2 Teilpfaden die Anfrageregion überdeckt. Da dieser Wert auch die untere Schranke ist, ist die Kurve optimal bzgl. dieser Aufgabe. Die Struktur der Kurve *(snake curve)* ist in Abbildung 2.17 skizziert.

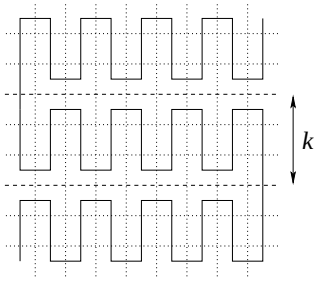


Abbildung 2.17: Optimale Kurve für feste $k \times k$-Anfrage

Für variable Anfragegrößen konnten sie die untere Schranke von drei Teilpfaden bei rekursiven raumfüllenden Kurven zeigen. Gleichzeitig konnten sie eine Kurve angeben, die diesen optimalen Wert erreicht. Für die graphische Definition der Kurve sind acht Verfeinerungsregeln notwendig. In Abbildung 2.18 ist die Kurve der Level $0, 1$ und 2 dargestellt. Sie besteht aus einem Mix von U- und N-förmigen Grundstrukturen, die in der Hilbert- und der Lebesgue-Kurve zu finden sind. Dadurch ist sichergestellt, dass jede 2×2-Anfrage mit höchstens drei Teilpfaden beantwortet werden kann. Die Autoren haben gezeigt, dass dieses auch für beliebige $k \times k$-Anfragen gilt, wenn eine Vergrößerung der Region um den Faktor 4 zugelassen wird.

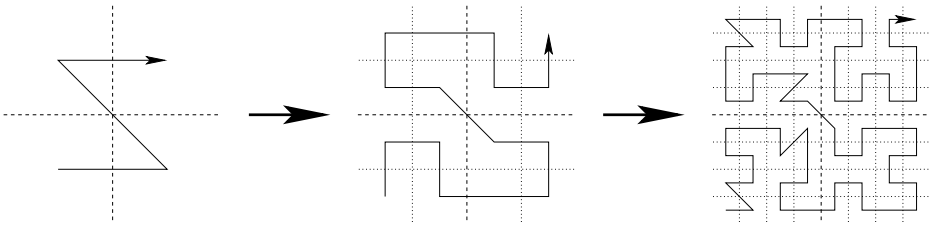


Abbildung 2.18: Anfrage-optimale raumfüllende Kurve

Die Eigenschaft der beschränkten Anzahl an Teilpfaden bei einer Anfrage im Raum wird für die Weiterentwicklung eines Algorithmus zur globalen Kontaktsuche in Kapitel 4 genutzt.[2] An der Stelle ist jedoch die Distanz zwischen den entstandenen Teilpfaden ebenfalls von großer Bedeutung, für die ein entsprechendes Maß definiert wird.

[2]Die in dieser Arbeit definierten Kurven werden durch eine $k \times k$-Anfrage in bis zu vier Teilepfade zerlegt. Dieser Fall tritt auf, wenn die Anfrage zwei Separatoren von höherem Level schneidet.

Kapitel 3

Partitionierung von Finite-Elemente-Netzen

Parallele Anwendungen auf Basis der Finite-Elemente-Methoden (FEM) basieren auf dem SPMD-Prinzip *(single program multiple data)*. Jeder Prozess arbeitet die gleichen Befehle ab, jedoch auf einem unterschiedlichen Bereich der Daten. Um eine hohe Skalierung der Applikation zu erreichen, sollte der Kommunikationsaufwand zwischen den nebenläufigen Prozessen möglichst gering sein. Die Datenabhängigkeiten können als Graph dargestellt werden, indem die Knoten die Daten und die Kanten deren Abhängigkeit im Verlauf der Rechnung repräsentieren. Insbesondere bei FE-Berechnungen ergeben sich in vielen aufeinander folgenden Schritten die gleichen Abhängigkeiten zwischen den Daten.

Bei der Aufteilung der Daten auf die Prozesse müssen Knoten so zu Gruppen *(Partitionen)* zusammengefasst werden, dass jede Partition die gleiche Rechenlast repräsentiert. Zudem sollen möglichst wenige Kanten zu Knoten außerhalb einer Partition bestehen, d. h. wenige Abhängigkeiten zu Daten anderer Prozesse. Die Struktur des Abhängigkeitsgraphen kann oftmals direkt vom gegebenen FE-Netz abgeleitet werden. Je nach Zuordnung der Daten zu den Knoten oder Elementen des Netzes, ist die Datenabhängigkeit häufig als Adjazenz der Knoten, der Elemente oder beider gegeben.

In diesem Kapitel wird die Eignung von raumfüllenden Kurven zur Partitionierung von FE-Netzen für die parallele numerische Simulation untersucht. Nach der Einführung der wesentlichen Grundlagen und einiger konkurrierender Methoden wird die Partitionierung mittels raumfüllender Kurven beschrieben. In den weiteren Abschnitten wird die Qualität in unterschiedlicher Hinsicht bewertet. Zum Einen werden analytische Ergebnisse zur absoluten Güte bei der Partitionierung regulärer Gitter dargestellt. Ein weiterer Abschnitt vergleicht die Qualität der vorgestellten Methode anhand eines Benchmark-Satzes aus unterschiedlichsten FE-Netzen mit den Ergebnissen der etablierten Partitionierungsbibliothek Metis. Zudem werden in Kapitel 5 Ergebnisse aus einer industriellen Anwendung aufgezeigt.

3.1 Grundlagen

Sei V eine Knotenmenge und $E \subseteq V \times V$ eine Menge an ungerichteten Kanten. Die Aufgabe der k-Partitionierung besteht in der Aufteilung des Graphen $G = (V, E)$ mit $n = |V|$ Knoten in k Teilgraphen V_i, mit

$$V_i \cap V_j = \emptyset \text{ , für } i \neq j \text{ und } i, j \leq k \quad \text{und} \quad \bigcup_{i=1}^{k} V_i = V \text{ ,} \tag{3.1}$$

so dass alle Teilgraphen gleich groß sind, d. h.

$$|V_i| - |V_j| \leq 1 \text{ , für } i \neq j \text{ und } i, j \leq k \tag{3.2}$$

Dabei soll der Kantenschnitt (engl.: *edge cut*) minimiert werden. Dieser umfasst alle Kanten E von G, die zwei Teilgraphen verbinden:

$$\text{edge cut} = |\{(u, v) \in E \,|\, u \in V_i,\, v \in V_j \text{ und } i \neq j\}|$$

Die vorgestellte Bewertung spiegelt die notwendigen Kriterien an eine Partitionierung nicht ausreichend wider. Es sind andere Metriken diskutiert worden, die besser den Bedürfnissen der parallelen Anwendungen entsprechen [36, 78]. Dennoch ist die vorgestellte Metrik der Quasi-Standard für die Bewertung von Methoden zur Graphpartitionierung und deren Umsetzung. Sie wird im weiteren Verlauf des Kapitels als Qualitätskriterium herangezogen.

Sind die Rechenlasten der einzelnen Knoten unterschiedlich, ist eine gewichtete Partitionierung notwendig. Dabei sollen die Rechenlasten gleichmäßig auf die Partitionen aufgeteilt werden. Sei $w(v)$ die Rechenlast des Knotens v für alle $v \in V$. Die gesamte Rechenlast ergibt sich zu

$$W = \sum_{v \in V} w(v) \text{ .}$$

Die gewichtete k-Partitionierung ist beschrieben durch:

$$\frac{W}{k} \cdot (1 - \varepsilon) \leq \sum_{v \in V_i} w(v) \leq \frac{W}{k} \cdot (1 + \varepsilon) \text{ , für } i = 1, \ldots, k \text{ .} \tag{3.3}$$

Dabei ist ε das zulässige Ungleichgewicht der Partitionierung.

Das Problem der Graphpartitionierung ist NP-vollständig, selbst für eine asymptotisch optimale Lösung [13]. Daher ist es i. A. nicht möglich, in annehmbarer Zeit eine optimale k-Partitionierung zu bestimmen. Diese Tatsache hat zu der Entwicklung einer großen Anzahl unterschiedlicher Heuristiken geführt. In mehreren Veröffentlichungen sind die bedeutendsten Verfahren beschrieben, klassifiziert und bewertet [18, 25, 35, 73].

Die Klasse der *geometrischen* Ansätze umfasst die schnellsten Verfahren. Zu diesen zählen *Recursive Coordinate Bisection* (RCB) und *Recursive Inertial Bisection* (RIB). Beide teilen die Knoten des Graphen durch gerade Schnitte in zwei Teile, die wiederum

rekursiv weiter zerlegt werden. RCB führt lediglich achsenparallele Schnitte aus [10]. Die Schnitte beim RIB liegen hingegen orthogonal zu der Hauptträgheitsachse (engl.: *principal inertial axis*) [57]. Unter der Annahme, dass das Netz in dieser Richtung eine geringere Ausdehnung hat, ergibt sich ein geringerer Kantenschnitt. Da diese Verfahren die Struktur des Graphen unberücksichtigt lassen, ist die Qualität der resultierenden Partitionierungen nur unterdurchschnittlich [35]. Es ist leicht Beispiele zu finden, deren k-Partitionierung mittels dieser geometrischen Verfahren zu einem sehr hohen Kantenschnitt führt.

Die Partitionierung über raumfüllende Kurven zählt ebenso zu der Klasse der geometrischen Methoden. Ihre Eigenschaften hinsichtlich Partitionierungsqualität und Geschwindigkeit sind vergleichbar mit den anderen Methoden dieser Klasse [35]. Der Einsatz dieser Methode erlaubt in vielen parallelen Applikationen eine ausreichende Skalierbarkeit [58, 61, 64]. Der besondere Vorteil liegt jedoch in der einfachen Repartitionierung bei einer Änderung der Lastsituation. Die raumfüllenden Kurven definieren eine eindimensionale Ordnung über alle Knoten oder Elemente des Netzes. Die Lastbalancierung erfolgt durch eine Verschiebung der Intervallgrenzen, über welche die Partitionierung gegeben ist. Damit erfolgt die Bestimmung der Lastverteilung sehr schnell, wobei eine gleich bleibende Qualität der Partitionierung angenommen werden kann.

Die besten Partitionierungsergebnisse werden von mehrstufigen Verfahren erzielt (engl.: *multilevel schemes*). Diese vergröbern den Graphen in mehreren Schritten, bis er so wenige Knoten hat, dass eine optimale Partitionierung bestimmt werden kann. Es werden Gruppen von Knoten zu einem Metaknoten zusammengefasst, wenn diese viele innere und wenige äußere Kanten besitzen. Von diesen kann angenommen werden, dass sie bei einer guten Partitionierung gemeinsam innerhalb einer Partition liegen werden. Während der nachfolgenden schrittweisen Verfeinerung erben die Knoten die Partitionszugehörigkeit ihres Stellvertreters im jeweils gröberen Graphen. Nach jedem Schritt der Verfeinerung wird mittels lokaler Verbesserungsheuristiken die Partitionierung des aktuellen Graphen verbessert [22, 24, 47]. Hierarchische Verfahren zur Graphpartitionierung sind in verschiedenen Bibliotheken implementiert, wie Metis [45], Jostle [79], Chaco [37] und Party [66, 68].

3.2 Implizite Partitionierung

In einigen Applikationen ist die Qualität einer Partitionierung nicht so bedeutend wie die ständig gleichmäßige Verteilung der vorhandenen Rechenlast. Bei sich regelmäßig ändernden Lastsituationen ist damit eine häufige Umverteilung der Last unumgänglich. Damit wächst der Einfluss der Zeiten für die Berechnung der jeweiligen Partitionierungen auf die Gesamtrechenzeit. Die *implizite* Partitionierung ist bei diesen speziellen Anforderungen besonders geeignet. Bei dieser wird vor der eigentlichen Berechnung eine eindimensionale Ordnung über die Last (hier: Knoten des Graphen) definiert, so dass sich eine Partitionierung direkt aus der Aufsplittung in Intervalle ergibt.

3.2.1 Datenverteilung

Die Aufgabe der Partitionierung kann durch die Sortierung der Knoten und deren anschließender Zerlegung in konsekutive Intervalle ersetzt werden. Unabhängig von der Sortierung $V = (v_1, \ldots, v_n)$ der Knoten erfüllt eine gleichmäßige Aufteilung die Bedingungen in (3.1) und (3.2) für eine k-Partitionierung:

$$V_i := \left\{ v_j \,\middle|\, \frac{(i-1)\cdot n}{k} < j \leq \frac{i \cdot n}{k} \right\} \text{ , für } 1 \leq i \leq k \text{ .} \tag{3.4}$$

Das gleiche gilt für die gewichtete Partitionierung in (3.3), jedoch mit der Einschränkung, dass die Ausgewogenheit der Partitionen durch die vorgegebene Reihenfolge begrenzt ist. Es ist jedoch offensichtlich, dass die Qualität der späteren Partitionierung wesentlich von der Ordnung der Knoten abhängig ist. Dieser Zusammenhang wird im weiteren Verlauf des Kapitels untersucht.

3.2.2 Grundlegende Eigenschaften

In Abbildung 3.1 sind die untersuchten Kurven gemäß Hilbert-, Lebesgue-, Sierpiński- und $\beta\Omega$-Indizierung dargestellt. In der oberen Reihe ist die jeweilige Kurve nach dem vierten Verfeinerungsschritt dargestellt. In der unteren Reihe ist die resultierende 5-Partitionierung des 16×16-Gitters angegeben.

Einerseits lassen sich aus der Partitionierung des Gitters keine allgemeinen Aussagen über die Partitionierungsqualität von raumfüllenden Kurven herleiten. Auf der anderen Seite ermöglicht die reguläre Struktur des Netzes, die Form der abgeleiteten Partitionen zu analysieren und Schwächen dieser geometrischen Heuristik auszumachen. Im 16×16-Gitter ergeben sich Kantenschnitte von 65 (Hilbert), 64 (Lebesgue), 66 (Sierpiński) und 58 ($\beta\Omega$).

Die Partitionierungsbibliothek Metis erzielt einen Kantenschnitt von 63 bei der *direkten* k-Partitionierung (kMetis) und 46 bei der Partitionierung mittels *rekursiver* Bisektion (pMetis). Die entsprechenden Partitionierungen sind in Abbildung 3.2 angegeben. Es ist offensichtlich, dass diese Aufgabe kMetis Probleme bereitet. Der Kantenschnitt zwischen der blauen und der gelben Partition kann um 14 verringert werden, in dem einige Knoten zwischen den Partitionen ausgetauscht werden. Bei Ausklammerung des Fehlers an der gelben Partition wird ein Kantenschnitt von 49 erreicht. Im Vergleich zu der von pMetis gelieferten Lösung haben die auf raumfüllenden Kurven basierenden Partitionierungen in Abbildung 3.1 einen um 29 bis 44 Prozent höheren Kantenschnitt.

Ein wesentlicher Nachteil der Partitionierung auf der Basis von raumfüllenden Kurven liegt in deren gewundener Struktur. An den Intervallgrenzen im Indizierungsschema, welche die Partitionen begrenzen, ergibt sich in vielen Fällen ein schneckenförmiger Ansatz. Dieser erzeugt einen, im Verhältnis zu der Anzahl der enthaltenen Knoten, sehr großen Kantenschnitt. Am deutlichsten zeigt sich dieser Nachteil bei der Hilbert-Kurve, an der Grenze zwischen der roten und der gelben Partition. Das gleiche Verhalten zeigen auch die Sierpiński- und $\beta\Omega$-Kurve, jedoch in leicht abgeschwächter Form.

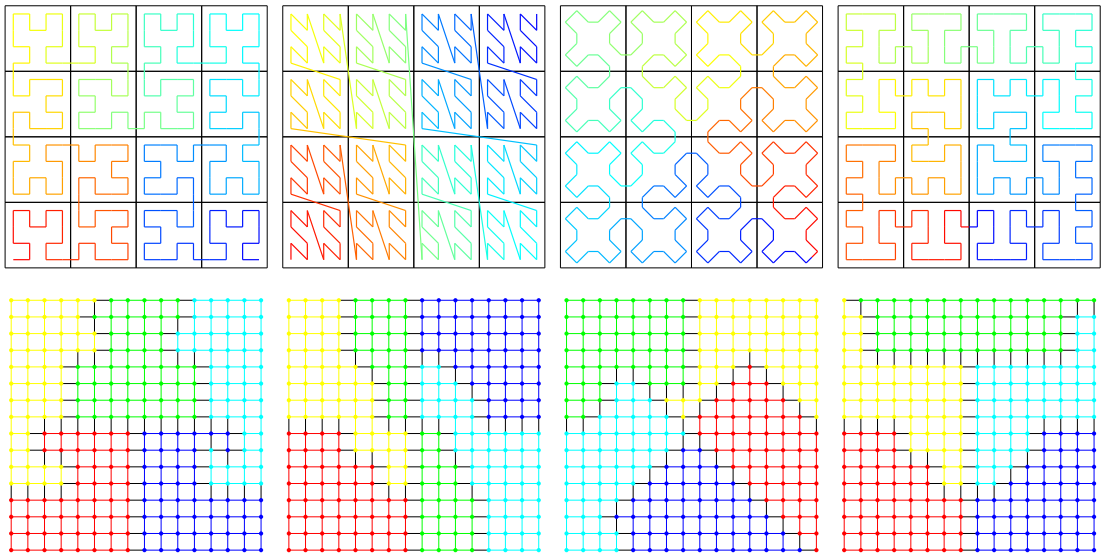


Abbildung 3.1: Partitionierungen des 16 × 16-Gitters

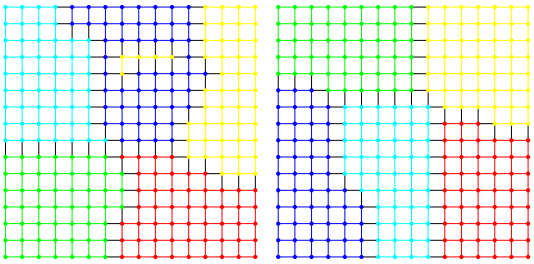


Abbildung 3.2: 5-Partitionierung des 16 × 16-Gitters durch Metis

Ein bedeutender Nachteil der Sierpiński-Kurve liegt jedoch in den diagonalen Separationslinien. Da die Partitionsgrenzen den Separationslinien folgen, ergibt sich im Gitter ein hoher Kantenschnitt, auch wenn die Partitionen selbst von kompakter Struktur sind. Diese Auffälligkeit ist jedoch auf Gitter und Netze vergleichbarer Struktur beschränkt. Die Sprünge innerhalb der Lebesgue-Kurve wirken sich negativ auf die darauf basierende Partitionierung aus. Die Partitionen sind nicht zwangsläufig zusammenhängend, wodurch sich ein großer Kantenschnitt ergibt, wie an der grünen Partition zu sehen ist.

3.2.3 Raumfüllende Kurven in unstrukturierten Netzen

Die vorgestellten raumfüllenden Kurven sind nur für Gitter definiert. Für die Sortierung der Knoten eines unstrukturierten Netzes wird das Netz gemäß der Separationslinien soweit rekursiv aufgeteilt, bis jedes Teilgebiet höchstens einen Knoten enthält.

In Abbildung 3.3 ist ein möglicher Algorithmus für die Bestimmung der Hilbert-Ord-

nung in einem unstrukturierten Netz skizziert. Die HILBERT-Funktion separiert alle Knoten des Intervalls [*first*,*last*[in vier Teile, die rekursiv weiter verarbeitet werden. Die Separatoren für die Zerlegung in die vier Gebiete sind *firstcut*, *midcut* und *lastcut*. Die SPLIT-Operation sortiert die Knoten bzgl. der angegebenen Orientierung (*orientation*; x- oder y-Koordinaten) und Richtung (aufsteigend oder absteigend). Sie gibt den Index zurück, der das Intervall gemäß dem gegebenen Separator teilt. Orientierung und Richtung für diese Operation ergeben sich aus dem Typ der Basisstruktur (*type*), die der aktuellen Rekursion zugrunde liegen. Die zum Algorithmus passende Nummerierung der Basisstrukturen ist in der Abbildung 3.3 rechts angegeben.

HILBERT (*first*, *last*, *type*):

orient1 = (*type* < 2) ? x : y
orient2 = (*type* < 2) ? y : x
crit1 = (*type*%2==0) ? ascend : descend
crit2 = (*type*%2==0) ? descend : ascend

midcut = SPLIT (*crit1*, *orient1*, *first*, *last*)
firstcut = SPLIT (*crit2*, *orient1*, *first*, *midcut*)
thirdcut = SPLIT (*crit2*, *orient2*, *midcut*, *last*)

HILBERT (*first*, *firstcut*, (*type*+2)%4)
HILBERT (*firstcut*, *midcut*, *type*)
HILBERT (*midcut*, *thirdcut*, *type*)
HILBERT (*thirdcut*, *last*, 3−*type*)

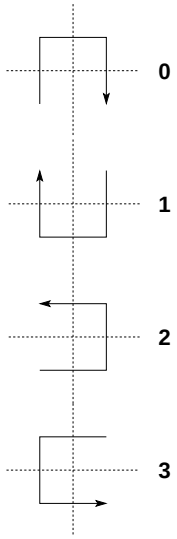


Abbildung 3.3: Algorithmen-Skizze zur Berechnung der Hilbert-Ordnung

Das Vorgehen des Algorithmus ist in Abbildung 3.4 an einem Beispiel dargestellt. Der Graph wird rekursiv gemäß der Separationslinien aufgeteilt. Die Ordnung der Knoten ergibt sich dabei aus der raumfüllenden Kurve vom jeweiligen Feinheitsgrad. Das linke Bild zeigt den ersten Schnitt. Alle Knoten im unteren linken Quadranten liegen vor

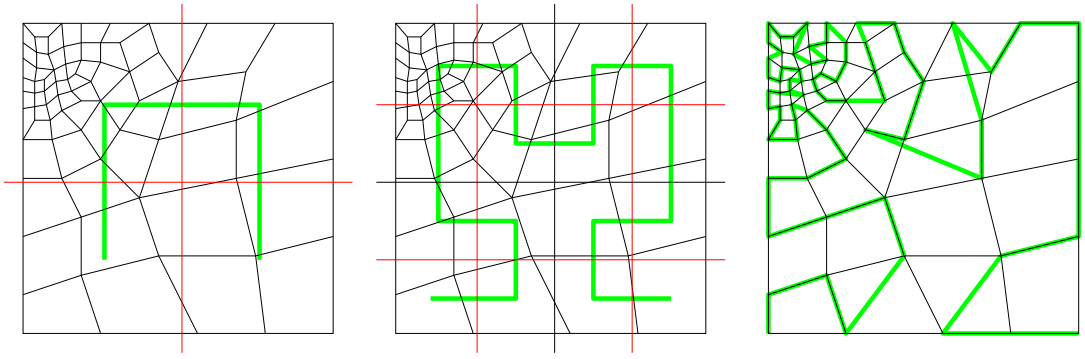


Abbildung 3.4: Knotensortierung gemäß Hilbert-Kurve in einem unstrukturierten Netz

den Knoten im oberen linken Quadranten usw. Das mittlere Bild zeigt den zweiten Rekursionsschritt. Die Rekursion wird solange fortgeführt, bis ein Teilbereich, der durch die Separationslinien abgegrenzt ist, höchstens einen Knoten enthält. Für diesen ist eine eindeutige Position in der Ordnung der Knoten gefunden. Für dieses Beispiel ist die Ordnung der Knoten durch die grüne Linie in der rechten Darstellung skizziert.

An dem gegebenen Beispiel zeigt sich, dass auch für die raumfüllenden Kurven, die im Gitter keine Sprünge aufweisen, in einem irregulären Graph i. A. keine Folge über adjazente Knoten entsteht. Es fließt nur die geometrische Position, nicht aber die Nachbarschaften der Knoten in die Partitionierung ein. Damit können stark verzerrte Netze zu deutlich schlechteren Partitionierungen führen. Dieser Effekt verstärkt sich, wenn das Netz Löcher enthält. Somit lassen sich für raumfüllende Kurven, ähnlich wie für jede Heuristik, die lediglich auf geometrischen Informationen beruht, Beispiele finden, in denen eine sehr schlechte Lösung erzielt wird.

Bei Netzen mit stark unterschiedlichen Ausdehnungen in x- und y-Richtung wird sich bei der rekursiven Generierung einer raumfüllenden Kurve eine stark verzerrte Kurve ergeben. Unter der Annahme, dass das Netz nicht verzerrte Elemente enthält, ergibt sich dabei eine schlechtere Partitionierungsqualität der Indizierung. Daher ist es sinnvoll, zunächst das Gebiet rekursiv bzgl. der längsten Ausdehnung zu teilen, bis das Verhältnis der Gebietsausdehnung kleiner oder gleich $\sqrt{2}$ ist. Dies ist das beste Verhältnis, das im schlechtesten Fall erreicht werden kann. Ein Beispiel für die Hilbert-Kurve ist in Abbildung 3.5 angegeben.

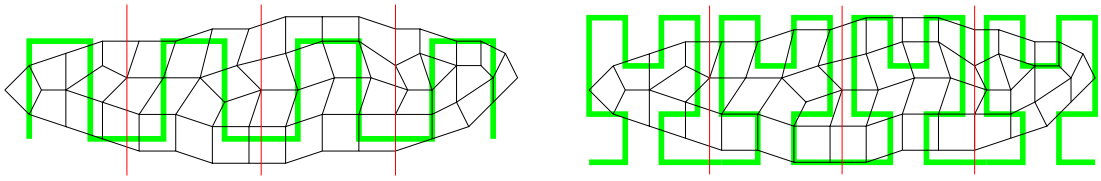
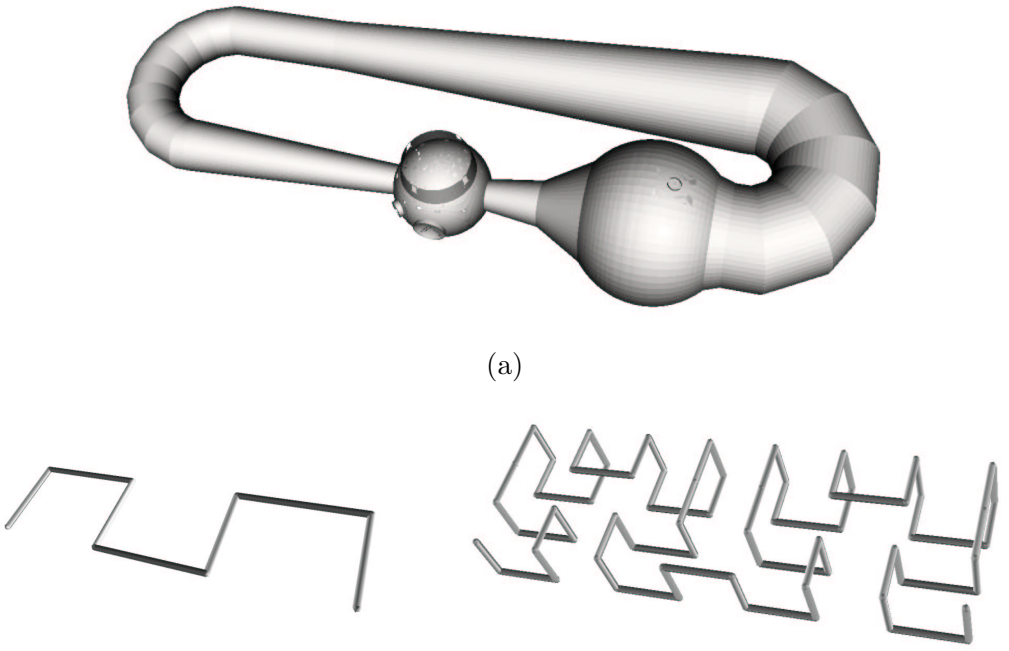


Abbildung 3.5: Erste Schritte zur Indizierung eines länglichen Gebietes bzgl. der Hilbert-Kurve

Zunächst wird das Gebiet in der Orientierung mit der größten Ausdehnung mittels Separatoren zerlegt, so dass das gewünschte Verhältnis der Ausdehnung erreicht ist. In dem gegebenen Beispiel beträgt das Verhältnis $22:6$. Nach der ersten Teilung mit dem mittleren Separator ergibt sich jeweils eine Ausdehnung von $11:6$, so dass ein weiterer Schnitt erfolgt (links). In jedem der vier Teilgebiete erfolgt die Zerlegung gemäß der Hilbert-Indizierung. Begonnen wird mit einer der Grundstrukturen, die eine Verkettung in der geforderten Orientierung erlaubt (vgl. Abschnitt 2.1.1). Sowohl für die x- als auch für die y-dominierte Ausdehnung ergeben sich zwei Möglichkeiten. Das rechte Bild zeigt den Beispielgraphen nach dem zweiten Rekursionsschritt der zweidimensionalen Hilbert-Indizierung.

Die gleiche Strategie lässt sich auf die Indizierung dreidimensionaler Netze übertragen. In diesem Fall lässt sich das Verhältnis der größten zur kleinsten Ausdehnung eines

Teilbereiches auf $2^{2/3}$ reduzieren.[1] In Abbildung 3.6 (a) ist die Oberfläche des Netzes *pwt* (vgl. Abschnitt 3.4.2) dargestellt. Zunächst wird das Gebiet bzgl. der x-Orientierung geteilt. Im zweiten Schritt werden beide Teile bzgl. der zweidimensionalen Hilbert-Kurve in x- und y-Richtung geteilt. Abbildung 3.6 (b) zeigt die Kurve nach dem zweiten Schritt. Danach beginnt die rekursive Aufteilung gemäß der dreidimensionalen Hilbert-Kurve. In 3.6 (c) ist die Kurve nach der ersten dreidimensionalen Aufteilung dargestellt. Es muss wiederum beachtet werden, dass eine Grundstruktur gewählt wird, die eine durchgehende Verkettung der Teilkurven erlaubt.



(a)

(b) (c)

Abbildung 3.6: Erste Schritte zur Indizierung des Netzes *pwt* bzgl. der dreidimensionalen Hilbert-Kurve

Es ergibt sich eine große Anzahl an Basisstrukturen geringerer Dimension und ebenso viele Ableitungsmöglichkeiten in den Raum mit einer höheren Dimension. Um die Anzahl der zu berücksichtigenden Fälle gering zu halten, ist es sinnvoll, das Objekt zunächst so zu drehen, dass die größte Ausdehnung in x-Richtung und die zweitgrößte Ausdehnung in y-Richtung liegt. Dann erfolgt zunächst eine eindimensionale Aufteilung bzgl. der x-Richtung, bis das gewünschte $\sqrt{2}:1$ Verhältnis erreicht ist. Die folgende zweidimensionale Zerlegung findet in xy-Richtung statt, wobei eine Grundstruktur mit x-Ausrichtung eingesetzt wird (vgl. Abb. 3.6 (b)). Bei der zweidimensionalen Zerlegung ergeben sich vier verschiedene Grundstrukturen. Beim Übergang zu der dreidimensio-

[1] Im schlechtesten Fall liegen die drei Ausdehnungen (o.B.d.A.) $a < b < c$ jeweils um den Faktor $2^{2/3}$ auseinander. Halbiert man die größte Ausdehnung, ändert sich die Ordnung in $c/2 < a < b$, die Abstände bleiben jedoch erhalten.

nalen Zerlegung, müssen diese vier Strukturen durch entsprechende dreidimensionale ersetzt werden. Um die durchgehende Verkettung zu gewährleisten, müssen die Projektionen der dreidimensionalen Grundstrukturen auf die xy-Ebene die gleichen Endpunkte haben, wie die zweidimensionalen, denen sie entsprechen. Zudem müssen die Endpunkte in der gleichen Ebene liegen. Abbildung 3.6 (c) zeigt eine mögliche Wahl von geeigneten dreidimensionalen Grundstrukturen. Die Endpunkte liegen alle in der oberen xy-Ebene.

3.3 Partitionierungsqualität im Gitter

Dieser Abschnitt beschreibt analytische Untersuchungen zur Qualität der Partitionen, die durch verschiedene Indizierungsschemata erzeugt werden. Die Ergebnisse basieren auf dem zweidimensionalen Gitter der Größe $2^n \times 2^n$ mit $n \in \mathbb{N}$, in das die raumfüllenden Kurven direkt eingebettet werden können. Anstelle der üblichen Partitionierung der Knoten eines Graphen wird in diesem Abschnitt aufgrund der besseren Darstellbarkeit die Partitionierung der Elemente untersucht. Das Gitter der Größe $2^n \times 2^n$ Knoten ist jedoch dual zu dem Netz aus $2^n \times 2^n$ Elementen, womit sich die Ergebnisse direkt auf die übliche Graphenpartitionierung übertragen lassen. Die Definition des Qualitätsmaßes ist angelehnt an die Arbeiten von Zumbusch [88].

Die folgende Analyse betrachtet die Partitionierungsqualität der Kurven Hilbert, Lebesgue und $\beta\Omega$, wobei sowohl der schlechteste als auch der mittlere Fall untersucht werden. Ein wesentliches Ergebnis der Analyse des schlechtesten Falls ist der Beweis, dass die Partitionierungsqualität der Lebesgue-Kurve besser ist, als die der beiden anderen untersuchten Indizierungsschemata.

In der Literatur ist es allgemein üblich, Partitionierungsverfahren über die Qualität der gesamten Partitionierung zu bewerten und zu vergleichen. Aufgrund der Besonderheiten bei der impliziten Partitionierung ist es an dieser Stelle möglich, die Qualität einer einzelnen Partition zu ermitteln.

3.3.1 Definition des Qualitätsmaßes

Das in diesem Abschnitt untersuchte Qualitätsmaß ist angelehnt an die Eigenschaften von geometrischen Körpern. Je nach Struktur eines Körpers ergibt sich eine bestimmte Abhängigkeit der Oberfläche zu seinem Volumen. In diesem Fall soll die Oberfläche, d. h. der Rand der Partition minimiert werden. Im zweidimensionalen Raum weist der Kreis, im dreidimensionalen Raum die Kugel die kleinste Oberfläche zu einem gegebenen Volumen auf. Die Kreisoberfläche wächst proportional zur Wurzel der Fläche, die Kugeloberfläche proportional zur dritten Wurzel des Volumens. Ein ähnliches Verhältnis von Oberfläche zu Volumen sollte i. A. auch die optimale Partitionierung von zwei- oder dreidimensionalen FE-Netzen aufweisen.

Für die in diesem Abschnitt untersuchten zweidimensionalen Gitter hat Zumbusch [88] den Vergleich mit dem Oberflächen-Volumen-Verhältnis des Quadrates vorgeschlagen,

welches diesbezüglich im Gitter optimale Eigenschaften aufweist. In seinen Arbeiten ist der schlechteste Fall für die Partitionierung mit raumfüllenden Kurven asymptotisch untersucht. Definition 3.1 ist eine Erweiterung des Qualitätsmaßes auch für den mittleren Fall.

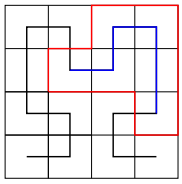


Abbildung 3.7: Oberfläche einer Partition in der Hilbert-Kurve

Abbildung 3.7 zeigt an der Hilbert-Kurve im 4×4-Gitter die Oberfläche einer Partition (blau) mit Volumen 6. Es ergibt sich eine Oberfläche (rot) von 12. Diese Form der Untersuchung schließt auch den äußeren Rand des Gitters mit ein. Der relative Anteil dieses Randes wird mit wachsender Größe des Gitters immer geringer, so dass er auf die unten definierte Partitionierungsqualität keinen Einfluss hat.

Definition 3.1 (Partitionierungsqualität) *Sei curve eine diskrete raumfüllende Kurve. Jedes Indexintervall $p = [i_1, i_2]$, mit $i_1 \leq i_2$ für $i_1, i_2 \in \mathbb{N}$, beschreibt eine Partition der Größe $V(p) = |p| = i_2 - i_1 + 1$. Sei $S(p)$ der Rand dieser Partition. Die Qualität der Partition ist gegeben als $\mathcal{P}^{curve}(p)$ mit:*

$$\mathcal{P}^{curve}(p) = \frac{S^{curve}(p)}{4 \cdot \sqrt{V(p)}} \tag{3.5}$$

Die Verallgemeinerung des Qualitätskriteriums für eine Kurve curve im schlechtesten Fall ergibt sich als:

$$\mathcal{P}^{curve}_{\max} = \max\{\mathcal{P}^{curve}(p) \,|\, p = [i_1, i_2] \text{ mit } i_1 \leq i_2 \text{ für } i_1, i_2 \in \mathbb{N}\} \tag{3.6}$$

Sei $i \in \mathbb{N}$ eine Intervalllänge. Die Qualität im mittleren Fall für eine gegebene Partitionsgröße ist:

$$\mathcal{P}^{curve}_{\text{avg}}(i) = \text{avg}\{\mathcal{P}^{curve}(p) \,|\, p = [i_1, i_2] \text{ mit } i_2 - i_1 + 1 = i \text{ für } i_1, i_2 \in \mathbb{N}\}, i \in \mathbb{N}\ , \tag{3.7}$$

wobei avg *das arithmetische Mittel ist.*

Die Verallgemeinerung für die mittlere Qualität einer Familie von Kurven curve ergibt sich als:

$$\mathcal{P}^{curve}_{\text{avg}} = \max\{\mathcal{P}^{curve}_{\text{avg}}(i) \,|\, i \in \mathbb{N}\} \tag{3.8}$$

In seinen Arbeiten hat Zumbusch gezeigt, dass kontinuierliche raumfüllende Kurven zu einer „quasi-optimalen“ Partitionierung des Gitters führen [88]. Für diese Kurven existiert eine Konstante C_{part}, so dass im r-dimensionalen Gitter für eine Partition der Größe v die obere Schranke

$$s \leq C_{\text{part}} \cdot v^{(d-1)/d} \tag{3.9}$$

für die Oberfläche der Partition gilt. Dabei ist die Konstante C_{part} von der Kurve abhängig und entspricht im zweidimensionalen Gitter $4 \cdot \mathcal{P}_{\max}^{curve}$ aus der obigen Definition. Die Begründung für die Schranke erfolgt über die Ergebnisse zur Lokalität kontinuierlicher raumfüllender Kurven (vgl. Abschnitt 2.2.1).

Zudem konnte er zeigen, dass die obere Schranke (3.9) auch für einige irreguläre Graphen gilt, wenn diese mittels schrittweiser Verfeinerung aus einem Gitter entstanden sind [87]. Für diese als „quasi-uniform“ bezeichnete Struktur müssen in jedem Verfeinerungsschritt bestimmte Schranken eingehalten werden, die ein Intervall für die Anzahl der Elemente angeben, in die ein vorhergehendes Element zerlegt werden muss. Wird eine Kurve durch die Elemente gelegt, deren Struktur in den Verfeinerungsschritten sukzessive erweitert wird („verallgemeinerte Sierpiński Kurve“), kann die oben angegebene Schranke für die Oberfläche erzielt werden.

In den folgenden Abschnitten werden Schranken für die Partitionierungsqualität im zweidimensionalen Gitter hergeleitet. Neben dem schlechtesten Fall wird dabei auch der mittlere Fall betrachtet, der für die Partitionierung eines Graphen in viele Teile von größerer Bedeutung ist.

3.3.2 Qualität im schlechtesten Fall

Zunächst wird die Partitionierungsqualität raumfüllender Kurven im schlechtesten Fall untersucht. Für die Lebesgue-Kurve kann über eine obere und eine untere Schranke eine genaue Abschätzung der Qualität angegeben werden. Für die Qualität der Hilbert- und der $\beta\Omega$-Kurve können untere Schranken gezeigt werden, die beweisen, dass die Lebesgue-Kurve bei der Betrachtung des schlechtesten Falles die beste Partitionierungsqualität der untersuchten Kurven aufweist. Experimentelle Ergebnisse in Abschnitt 3.3.4 belegen, dass die gefundenen unteren Schranken wahrscheinlich sehr nahe an den tatsächlichen Partitionierungsqualitäten der beiden Kurven liegen.

3.3.2.1 Lebesgue-Kurve

Die analytischen Untersuchungen beginnen mit der Lebesgue-Kurve. Diese ist durch ihre einfache Konstruktionsvorschrift, ohne Spiegelung oder Drehung der Basisstruktur, am leichtesten zu analysieren. Daher ist es insbesondere möglich, auch eine relativ scharfe obere Schranke für die Qualität zu bestimmen.

Untere Schranke Der Beweis der unteren Schranke erfolgt durch die Konstruktion einer Partition entlang der Lebesgue-Kurve mit schlechter Qualität. Die symmetrische Partition ist durch eine Separationslinie von hohem Level geteilt. Jede Hälfte enthält Quadrate der Größe 4^k, 4^{k-1}, 4^{k-2}, …, 4^2, 4^1 und zwei der Größe 4^0. Eine Hälfte der Partition und deren Lage innerhalb der gesamten Kurve ist in Abbildung 3.8 dargestellt.

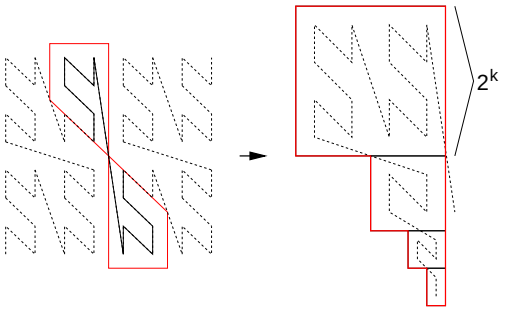


Abbildung 3.8: Untere Schranke für die Partitionierungsqualität der Lebesgue-Kurve

Satz 3.1 *Für die Partitionierungsqualität der Lebesgue-Kurve gilt im schlechtesten Fall*

$$\mathcal{P}_{\max}^{\text{Lebesgue}} \geq 3\sqrt{\frac{3}{8}} \ .$$

Beweis: Für jede Hälfte der Partition p_k ergibt sich das Volumen V zu

$$V(p_k) = \sum_{i=0}^{k} 4^i + 4^0 = \frac{4^{k+1}-1}{3} + 1 = \frac{4^{k+1}+2}{3} \tag{3.10}$$

und die Oberfläche S zu

$$S(p_k) = 6 \cdot 2^k \ . \tag{3.11}$$

Aus (3.10) und (3.11) folgt die Qualität $\mathcal{P}$ der gesamten Partition als

$$\begin{aligned}
\mathcal{P}^{\text{Lebesgue}}(p_k) &= \frac{2 \cdot S(p_k)}{4 \cdot \sqrt{2 \cdot V(p_k)}} \\
&= \frac{1}{2\sqrt{2}} \cdot \frac{6 \cdot 2^k}{\sqrt{\frac{4^{k+1}+2}{3}}} \\
&= 3\sqrt{\frac{3}{2}} \cdot \frac{2^k}{\sqrt{4^{k+1}+2}} \\
&> 3\sqrt{\frac{3}{2}} \cdot \frac{2^k}{2^{k+1}+1} \qquad \text{, da k} \geq 0 \\
&= 3\sqrt{\frac{3}{2}} \cdot \frac{2^k \cdot \left(2^{k+1}-1\right)}{2^{2k+2}-1} \\
&> 3\sqrt{\frac{3}{2}} \cdot \frac{2^{2k+1}-2^k}{2^{2k+2}} \\
&= 3\sqrt{\frac{3}{2}} \cdot \left(\frac{1}{2} - \frac{1}{2^{k+2}}\right)
\end{aligned} \tag{3.12}$$

Für große k konvergiert $\mathcal{P}^{\text{Lebesgue}}(p_k)$ gegen $3 \cdot \sqrt{\frac{3}{8}} \approx 1.84$. □

Obere Schranke Eine Partition, die sich aus der Struktur der Lebesgue-Kurve ergibt, kann in höchstens zwei unverbundene Gebiete zerfallen. Die Bestimmung einer oberen Schranke für die Partitionierungsqualität basiert auf der Betrachtung dieser beiden Teilpartitionen. Sie erfüllen besondere Eigenschaften, welche die Analyse erleichtern. Eine der beiden Teilpartitionen endet in der oberen rechten Ecke des Quadranten, der sie vollständig enthält, die andere beginnt in der unteren linken Ecke des Quadranten, der sie vollständig enthält (vgl. Abbildung 3.8). Damit sind beide Teilpartitionen bzgl. dieser Eigenschaft symmetrisch zueinander. Zudem ist die Oberfläche einer zusammenhängenden Partition im Falle der Lebesgue-Kurve kleiner oder gleich der Oberfläche eines umschließenden Rechtecks[2]. Letztere Eigenschaft erlaubt die Analyse grob-granularer Strukturen. Über diese ergibt sich für eine gegebene Partitionsgröße eine obere Schranke für die Oberfläche, woraus sich eine obere Schranke für die Partitionierungsqualität ableiten lässt.

Ein einführendes Lemma betrachtet zunächst die Qualität einer Partition, die mit dem ersten Gitterelement beginnt und damit in jedem Fall zusammenhängend ist.

Lemma 3.1 *Für jede Partition $p = [1, V]$, die sich aus der Lebesgue-Kurve ergibt, gilt*

$$\mathcal{P}_{\max}^{\text{Lebesgue}} \leq \frac{3}{\sqrt{5}} \ .$$

Beweis: Für eine gegebene Partition $p = [1, V]$ mit Volumen $V > 4$ wähle $k \in \mathbb{N}$, so dass $4 \cdot 4^k < V \leq 16 \cdot 4^k$. Für jedes V in diesem Intervall existiert ein v mit der Eigenschaft $v \cdot 4^k < V \leq (v+1) \cdot 4^k$. Die Oberfläche $S(p)$ von p ist kleiner oder gleich der Oberfläche der Partition $[1, (v+1) \cdot 4^k]$. Die folgende Tabelle zeigt obere Schranken für die Oberfläche für Partitionen mit Volumen $V \leq (v+1) \cdot 4^k$. Die Oberfläche ist in der untersuchten Granularität angegeben, d. h. s ist so gewählt, dass $s \cdot 2^k \geq S$ gilt[3].

v	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s	4	6	8	8	10	12	12	12	14	14	16	16	16	16	16	16

Für alle $V > 4$ gilt:

$$\begin{aligned} \frac{S(p)}{4 \cdot \sqrt{V(p)}} &\leq \max\left\{ \frac{s}{4 \cdot \sqrt{v}} \middle| \, 4 \leq v < 16 \right\} \\ &\leq \frac{3}{\sqrt{5}} \approx 1.34 \ , \end{aligned}$$

d. h. die schlechteste Qualität ergibt sich für $v = 5$.

Für kleine Partitionen mit weniger als 5 Elementen sind die Partitionierungsqualitäten 1.0 ($V = 1$), $1.5/\sqrt{2} \approx 1.061$ ($V = 2$), $2/\sqrt{3} \approx 1.155$ ($V = 3$), 1.0 ($V = 4$). □

[2] Diese Eigenschaft gilt nicht für alle untersuchten Kurven. Der schneckenförmige Verlauf bei Hilbert- und $\beta\Omega$-Kurve führt i. A. zu einer deutlich höheren Oberfläche.

[3] Die Tabelle enthält auch Werte für $v < 4$, die nicht in diesem Lemma betrachtet werden. Diese werden in Satz 3.2 verwendet.

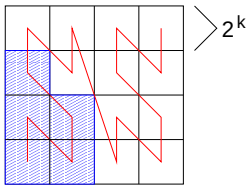


Abbildung 3.9: Partition der Form [1,V] in der Lebesgue-Kurve im 4×4-Gitter

Beispiel 3.1 Abbildung 3.9 zeigt das Oberflächen-Volumen Verhältnis für $v = 4$. Dieser Fall enthält alle Partitionen p der Form $[1, V]$ mit $4{\cdot}4^k < V \leq 5{\cdot}4^k$. Die resultierende Oberfläche S ist in jedem Fall kleiner oder gleich $10 \cdot 2^k$, d. h. es gilt $s = 10$. Daraus folgt eine Qualität der Partition von

$$\frac{10}{4\sqrt{4}} = \frac{5}{4}$$

im schlechtesten Fall.

Der gleiche Beweisansatz wird für die Bestimmung der Partitionsqualität beliebiger Partitionen eingesetzt. Diese zerfallen in höchstens zwei unabhängige Teilpartitionen, welche die gleiche Struktur aufweisen, wie die in Lemma 3.1 untersuchten Partitionen. Damit ergibt sich eine Partition p als Teilpartitionen p_1 und p_2 mit $p = p_1 \circ p_2$, die beide in der Lebesgue-Struktur eines groben Gitters untersucht werden.

Satz 3.2 *Für jedes Intervall p der Lebesgue-Kurve gilt die Partitionsqualität*

$$\mathcal{P}_{\max}^{\text{Lebesgue}} \leq \frac{7}{2\sqrt{3}} \ .$$

Beweis: Sei $k \in \mathbb{N}_0$ so gewählt, dass $4 \cdot 4^k < V \leq 16 \cdot 4^k$. Sei $p = p_1 \circ p_2$ eine Zerlegung der Partition p, so dass p_1 an der letzten Position einer Teilkurve vom Level $k+1$ endet und p_2 an der ersten Position einer Teilkurve vom Level $k+1$ endet (vgl. Voraussetzung in Lemma 3.1). Falls beide Partitionen geometrisch zusammenhängend sind, verringert sich die maximal mögliche Oberfläche und damit verbessert sich die Qualität der Gesamtpartition p. Daher wird im Folgenden angenommen, dass beide Teilpartitionen nicht verbunden sind.

Seien $V_1 = |p_1|$ und $V_2 = |p_2|$ die Volumen der Teilpartitionen. Es existieren v_1 und v_2 mit der Eigenschaft $v_{1,2} \cdot 4^k < V_{1,2} \leq (v_{1,2} + 1) \cdot 4^k$. Es gilt $3 \leq v_1 + v_2 \leq 15$ mit $v_1, v_2 \in [0, 15]$. Für die Partitionsqualität gilt

$$\mathcal{P}(p_1 \circ p_2) \leq \frac{s_1 + s_2}{4 \cdot \sqrt{v_1 + v_2}} \ . \tag{3.13}$$

Die Betrachtung aller möglichen Kombinationen für v_1 und v_2 zeigt, dass das Maximum für $v_1 = 1$ und $v_2 = 2$ angenommen wird. Die zugehörigen oberen Schranken für die Oberflächen sind $s_1 = 6$ und $s_2 = 8$ (vgl. Tabelle in Lemma 3.1) und es folgt

$$\mathcal{P}(p_1 \circ p_2) \leq \frac{6+8}{4 \cdot \sqrt{1+2}} = \frac{7}{2 \cdot \sqrt{3}} \approx 2.02 \ .$$

□

Die Analyse der oberen Schranke erfolgt als Aufzählung einer endlichen Menge von Zerlegungen einer Partition. Für diese möglichen Kombinationen aus Teilpartitionen wird jeweils eine obere Schranke der Qualität ermittelt und aus diesen Schranken die höchste bestimmt. Die obere Schranke kann gesenkt werden, indem die Analyse in ein feineres Gitter übertragen wird. Dies ist gleichbedeutend mit der Verschiebung des untersuchten Intervalls $4 \cdot 4^k < V \leq 16 \cdot 4^k$. Eine rechnerunterstützte Auswertung ergibt folgende Verbesserungen der oberen Schranke:

$$4 \cdot 4^k < V \leq 16 \cdot 4^k \quad \Rightarrow \quad \mathcal{P}_{\max}^{\text{Lebesgue}} \leq \frac{6+8}{4 \cdot \sqrt{1+2}} < 2.021 \tag{3.14}$$

$$16 \cdot 4^k < V \leq 64 \cdot 4^k \quad \Rightarrow \quad \mathcal{P}_{\max}^{\text{Lebesgue}} \leq \frac{24+24}{4 \cdot \sqrt{21+21}} < 1.852 \tag{3.15}$$

$$256 \cdot 4^k < V \leq 1024 \cdot 4^k \quad \Rightarrow \quad \mathcal{P}_{\max}^{\text{Lebesgue}} \leq \frac{192+192}{4 \cdot \sqrt{1365+1365}} < 1.838 \tag{3.16}$$

Korollar 3.3 *Die Partitionierungsqualität der Lebesgue-Kurve liegt in dem Bereich*

$$1.837 < 3\sqrt{\frac{3}{8}} \leq C_{\max}^{\text{Lebesgue}} \leq \frac{96}{\sqrt{2730}} < 1.838 \ . \tag{3.17}$$

3.3.2.2 Hilbert-Kurve

Satz 3.4 *Für die Partitionierungsqualität der Hilbert-Kurve gilt im schlechtesten Fall*

$$\mathcal{P}_{\max}^{\text{Hilbert}} \geq 3\sqrt{\frac{5}{13}} \ .$$

Beweis: Der Beweis erfolgt über die Konstruktion einer Partition p_k mit Volumen $V(p_k)$ und Oberfläche $S(p_k)$, welche diese schlechte Partitionsqualität hat: Sei $k \in \mathbb{N}$ gerade. Das Zentrum der Partition ist gegeben als Quadrat der Größe 4^{k+1}. An zwei Seiten des Quadrates werden jeweils 3 kleinere Quadrate der Größen 4^k, 4^{k-2}, ..., 4^2, 4^0 schneckenförmig angehängt. Abbildung 3.10 zeigt die Konstruktion der Partition und ihre Lage innerhalb der Hilbert-Kurve. Damit ergibt sich für das Volumen V zu

$$\begin{aligned} V(p_k) &= 4^{k+1} + 2\sum_{i=0}^{k/2} 3 \cdot 4^{2i} \\ &= 4^{k+1} + 6\frac{16^{k/2+1} - 1}{15} \\ &= \frac{52}{5}4^k - \frac{2}{5} \ . \end{aligned} \tag{3.18}$$

Der entsprechende Rand S ergibt

$$S(p_k) = 8 \cdot 2^k + 2\sum_{i=0}^{k/2} \left(-2^{2i} + 7 \cdot 2^{2i}\right)$$

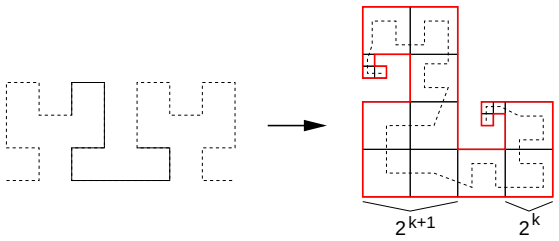


Abbildung 3.10: Untere Schranke für die Partitionierungsqualität der Hilbert-Kurve

$$\begin{aligned} &= 8 \cdot 2^k + 12 \sum_{i=0}^{k/2} 2^{2i} \\ &= 8 \cdot 2^k + 12 \frac{4^{k/2+1} - 1}{3} \\ &= 24 \cdot 2^k - 4 \ . \end{aligned} \tag{3.19}$$

Damit gilt für eine untere Schranke für Hilbert-Kurven

$$\begin{aligned} \mathcal{P}^{\text{Hilbert}}(p_k) &= \frac{S(p_k)}{4 \cdot \sqrt{V(p_k)}} \\ &= \frac{24 \cdot 2^k - 4}{4 \cdot \sqrt{\frac{52}{5} 4^k - \frac{2}{5}}} \\ &> \frac{24 \cdot 2^k - 4}{4\sqrt{\frac{52}{5}} 2^k} \\ &= \sqrt{\frac{5}{13}} \cdot \left(3 - \frac{1}{2^{k+1}} \right) \ . \end{aligned} \tag{3.20}$$

$\mathcal{P}^{\text{Hilbert}}(p_k)$ konvergiert gegen $3 \cdot \sqrt{\frac{5}{13}} \approx 1.86$ für große k. □

3.3.2.3 $\beta\Omega$-Indizierung

Satz 3.5 *Für die Partitionierungsqualität der $\beta\Omega$-Kurve gilt im schlechtesten Fall*

$$\mathcal{P}_{\max}^{\beta\Omega-\text{Ind}} \geq 3\sqrt{\frac{3}{7}} \ .$$

Beweis: Die Konstruktion einer Partition schlechter Qualität in der $\beta\Omega$-Kurve findet sich in der Verfeinerung der geraden Grundstruktur. Der Verlauf der Partition und die Lage innerhalb des Schemas sind in Abbildung 3.11 dargestellt. Im Zentrum der Partition p_k steht ein Quadrat der Größe 4^{k+1}. An zwei Seiten des Quadrates werden jeweils 2 kleiner Quadrate der Größen $4^k, \ldots, 4^1, 4^0$ schneckenförmig angehängt. Damit

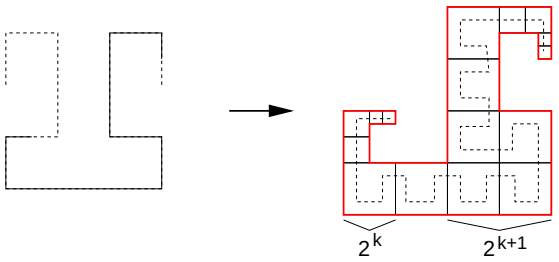


Abbildung 3.11: Untere Schranke für die Partitionierungsqualität der $\beta\Omega$-Kurve

gilt für das Volumen V

$$\begin{aligned} V(p_k) &= 4^{k+1} + 2\sum_{i=0}^{k} 2 \cdot 4^i \\ &= 4^{k+1} + 4\frac{4^{k+1}-1}{3} \\ &= \frac{28}{3}4^k - \frac{4}{3} \end{aligned} \tag{3.21}$$

und für den entsprechenden Rand S

$$\begin{aligned} S(p_k) &= 8 \cdot 2^k + 2\sum_{i=0}^{k} \left(-2^i + 5 \cdot 2^i\right) \\ &= 8 \cdot 2^k + 8\sum_{i=0}^{k} 2^i \\ &= 8 \cdot 2^k + 8\left(2^{k+1} - 1\right) \\ &= 24 \cdot 2^k - 8\ . \end{aligned} \tag{3.22}$$

Damit ergibt sich eine Partitionsqualität von

$$\begin{aligned} \mathcal{P}^{\beta\Omega-\mathrm{Ind}}(p_k) &\geq \frac{S(p_k)}{4 \cdot \sqrt{V(p_k)}} \\ &= \frac{24 \cdot 2^k - 8}{4\sqrt{\frac{28}{3}4^k - \frac{4}{3}}} \\ &> \frac{24 \cdot 2^k - 8}{8\sqrt{\frac{7}{3}2^k}} \\ &= \sqrt{\frac{3}{7}}\left(3 - \frac{8}{2^k}\right)\ . \end{aligned} \tag{3.23}$$

$\mathcal{P}^{\beta\Omega-\mathrm{Ind}}(p_k)$ konvergiert gegen $3\sqrt{\frac{3}{7}} \approx 1.96$ für große k. □

Korollar 3.6 *Im schlechtesten Fall ist die Partitionierungsqualität der Lebesgue-Kurve besser als die der Hilbert- und $\beta\Omega$-Kurve, d. h.*

$$\begin{aligned} \mathcal{P}^{\mathrm{Lebesgue}}_{\max} &< \mathcal{P}^{\mathrm{Hilbert}}_{\max} \quad \textit{und} \\ \mathcal{P}^{\mathrm{Lebesgue}}_{\max} &< \mathcal{P}^{\beta\Omega-\mathrm{Ind}}_{\max} \end{aligned}$$

Beweis: Der Beweis ergibt sich direkt aus den unteren Schranken von Hilbert- und $\beta\Omega$-Kurve (Sätze 3.4 und 3.5) und der oberen Schranke der Lebesgue-Kurve (Satz 3.3). □

3.3.3 Qualität der Lebesgue-Kurve im mittleren Fall

Dieser Abschnitt konzentriert sich auf die Untersuchung der Partitionierungsqualität im mittleren Fall. Aufgrund ihrer regelmäßigen Struktur lässt sich für die Lebesgue-Kurve eine sehr scharfe Schranke finden. Die Qualität wird über die Wahrscheinlichkeit bestimmt, dass eine benachbarte Zelle innerhalb der gleichen Partition liegt. Diese ist von der Größe der Partition, der relativen Position innerhalb der Partition und der Distanz über den Pfad der Kurve zur benachbarten Zelle abhängig. Eine Herleitung für die gleiche Schranke über die Aufzählung aller möglichen Fälle ist in Anhang A angegeben [80].

Eine Möglichkeit, die Größe der Oberfläche einer Partition zu berechnen, liegt in der Bestimmung aller Kanten, die zwischen zwei Zellen der Partition liegen (*innere* Kanten). Jede der V Gitterzellen hat 4 Kanten, wobei die inneren Kanten an jeweils zwei Zellen liegen. Damit ergibt sich die gesuchte Oberfläche aus der Differenz zwischen $4V$ und dem doppelten der Anzahl aller inneren Kanten. Eine Zelle hat eine innere Kante an der oberen oder rechten Seite, wenn der Index der Zelle oberhalb bzw. rechts klein genug ist, so dass sie auch noch Teil der gleichen Partition ist. Sei die obere Separationslinie einer Zelle vom Level q und die rechte vom Level p (vgl. Abb. 3.12). Dann sind die Indizes der oberen Zelle um $U_q = \frac{2\cdot 4^q+1}{3}$ und die rechten um $R_p = \frac{4\cdot 4^p+2}{3}$ größer (vgl. Lemma 2.1, S. 10).

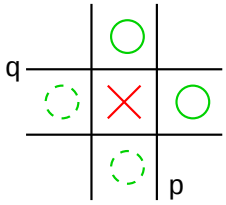


Abbildung 3.12: Nachbarschaft über Separationslinien beliebigen Levels an einer Zelle

Lemma 3.2 *Sei V die gewünschte Partitionsgröße. Im mittleren Fall ergibt sich bei der Lebesgue-Kurve eine Oberfläche S von*

$$S \leq \frac{3}{2^k}V + \frac{8}{3}2^k - \frac{5}{3}\frac{1}{2^k} \quad \text{für } V \in \left[\frac{2\cdot 4^k+1}{3}, \frac{4\cdot 4^k+2}{3}\right[\text{ und}$$

$$S \leq \frac{2}{2^k}V + 4\cdot 2^k - \frac{1}{2^k} \quad \text{für } V \in \left[\frac{4\cdot 4^k+2}{3}, \frac{2\cdot 4^{k+1}+1}{3}\right[, \text{mit } k \in \mathbb{N}_0 \ .$$

Beweis:

Die Anzahl der Zellen mit einer Nachbarzelle, die innerhalb der Partition liegt und durch eine Separationslinie vom Level l getrennt ist, ist gegeben durch $\max\{V - R_l, 0\}$

und $\max\{V-U_l,0\}$ für rechte und obere Nachbarn. Diese Anzahl ergibt zusammen mit der Wahrscheinlichkeit eines Levels l eine Oberfläche von

$$S \leq 4V - 2\sum_{i=0}^{\infty} \frac{1}{2^{i+1}} \max\{V-U_i,0\} - 2\sum_{i=0}^{\infty} \frac{1}{2^{i+1}} \max\{V-R_i,0\} \ . \tag{3.24}$$

Für die weitere Auswertung dieses Terms wird der betrachtete Raum in Intervalle der Klassen

$$\begin{aligned} I_1 &= \left[\frac{2\cdot 4^k+1}{3}, \frac{4\cdot 4^k+2}{3}\right[\quad \text{und} \\ I_2 &= \left[\frac{4\cdot 4^k+2}{3}, \frac{2\cdot 4^{k+1}+1}{3}\right[\end{aligned}$$

aufgeteilt. Die Größe der Oberfläche ist für alle Partitionen p in Intervallen der Klasse I_1

$$\begin{aligned} S_1 &\leq 4V - \sum_{i=0}^{k} \frac{1}{2^i}\left(V - \frac{2\cdot 4^i+1}{3}\right) - \sum_{i=0}^{k-1} \frac{1}{2^i}\left(V - \frac{4\cdot 4^i+2}{3}\right) \\ &= 4V - \sum_{i=0}^{k}\left(\frac{1}{2^i}\left(V-\frac{1}{3}\right) - \frac{2}{3}\cdot 2^i\right) - \sum_{i=0}^{k-1}\left(\frac{1}{2^i}\left(V-\frac{2}{3}\right) - \frac{4}{3}\cdot 2^i\right) \\ &= 4V - \left(V-\frac{1}{3}\right)\left(2-\frac{1}{2^k}\right) + \frac{2}{3}\left(2^{k+1}-1\right) \\ &\quad - \left(V-\frac{2}{3}\right)\left(2-\frac{1}{2^{k-1}}\right) + \frac{4}{3}\left(2^k-1\right) \\ &= \frac{3}{2^k}V + \frac{8}{3}2^k - \frac{5}{3}\frac{1}{2^k} \end{aligned} \tag{3.25}$$

und in Intervallen der Klasse I_2

$$\begin{aligned} S_2 &\leq 4V - \sum_{i=0}^{k} \frac{1}{2^i}\left(V - \frac{2\cdot 4^i+1}{3}\right) - \sum_{i=0}^{k} \frac{1}{2^i}\left(V - \frac{4\cdot 4^i+2}{3}\right) \\ &= 4V - \sum_{i=0}^{k}\left(\frac{1}{2^i}\left(V-\frac{1}{3}\right) - \frac{3}{2}\cdot 2^i\right) - \sum_{i=0}^{k}\left(\frac{1}{2^i}\left(V-\frac{2}{3}\right) - \frac{4}{3}\cdot 2^i\right) \\ &= 4V - \left(V-\frac{1}{3}\right)\left(2-\frac{1}{2^k}\right) + \frac{2}{3}\left(2^{k+1}-1\right) \\ &\quad - \left(V-\frac{2}{3}\right)\left(2-\frac{1}{2^k}\right) + \frac{4}{3}\left(2^{k+1}-1\right) \\ &= \frac{2}{2^k}V + 4\cdot 2^k - \frac{1}{2^k} \ . \end{aligned} \tag{3.26}$$

□

Anmerkung 3.1 An den Intervallgrenzen von I_1 bzw. I_2 sind die Werte für S_1 und S_2 identisch.

An dieser Stelle ist zu beachten, dass die Wahrscheinlichkeit für das Auftreten eines bestimmten Separators vom Level l nicht genau $p(l) = \frac{1}{2^{l+1}}$ ist. Für $l = 0$ ist die

Wahrscheinlichkeit größer als $p(0)$, für alle anderen Level ist die Wahrscheinlichkeit kleiner als $p(l)$. Damit gibt diese Berechnung eine untere Schranke der inneren Kanten einer Partition an und führt damit zu einer oberen Schranke für die Größe der Oberfläche. Die Analyse der exakten Schranke zeigt, dass die hier ermittelten Größen dem Konvergenzwert großer Gitter entsprechen. Entscheidend ist, dass die Qualität der angegebenen Schranken nur von der Größe des Gitters, *nicht* aber von der Größe der Partition abhängig ist.

Satz 3.7 *Für die Partitionierungsqualität der Lebesgue Kurve gilt im mittleren Fall die obere Schranke*

$$\mathcal{P}_{\text{avg}}^{\text{Lebesgue}} \leq \frac{5}{2\sqrt{3}} \ .$$

Beweis: Die schlechteste Partitionierungsqualität ergibt sich an den Rändern der Intervalle aus den Klassen I_1 und I_2. Für die Bestimmung einer oberen Schranke ist damit eine nähere Untersuchung dieser Stellen notwendig. Bezeichne a die Intervallgrenzen $\frac{2\cdot 4^k+1}{3}$ und b die Intervallgrenzen $\frac{4\cdot 4^k+2}{3}$. Seien S_a, S_b, V_a, V_b die Oberflächen und Volumina an den Stellen a und b. An den Positionen b (linke Ränder in der Intervall-Klasse I_2) gilt für das Volumen

$$\begin{aligned} & V_b &=& \frac{4\cdot 4^k+2}{3} \\ \Leftrightarrow \quad & 3V_b-2 &=& 4^{k+1} \\ \Leftrightarrow \quad & \frac{\sqrt{3V_b-2}}{2} &=& 2^k \text{ , da } V_b \in \mathbb{N}. \end{aligned} \tag{3.27}$$

Eingesetzt in das Resultat für S_2 aus Lemma 3.2 ergibt

$$\begin{aligned} S_b &\leq \frac{2}{2^k}V_b + 4\cdot 2^k - \frac{1}{2^k} \\ &= \frac{4V_b-2}{\sqrt{3V_b-2}} + 2\sqrt{3V_b-2} = \frac{10V_b-6}{\sqrt{3V_b-2}} \\ &\leq \frac{10}{\sqrt{3}}\sqrt{V_b} \quad \text{, da } V_b \in \mathbb{N} \end{aligned} \tag{3.28}$$

Für die Positionen a (linke Ränder der Intervalle in der Klasse I_1) gilt entsprechend

$$S_a \leq \frac{7\sqrt{2}}{\sqrt{3}} \cdot \sqrt{V_a} \ . \tag{3.29}$$

Die Oberfläche ist an der zunächst betrachteten Intervallgrenze (b) offensichtlich größer. Die Partitionierungsqualität im mittleren Fall ist damit gegeben durch

$$C_{\text{avg}}^{\text{Lebesgue}} \leq \frac{10}{\sqrt{3}} \cdot \frac{1}{4} = \frac{5}{2\sqrt{3}} \approx 1.44 \ . \tag{3.30}$$

□

Eine Analyse der Hilbert-Kurve ist in ähnlicher Form nicht möglich. Die Distanz über den Pfad der Kurve für zwei direkt benachbarte Zellen im Gitter ist nicht nur von dem Level des Separators abhängig. Abschnitt 4.5.3.2 zeigt, wie über die Untersuchung grobgranularer Strukturen eine obere Schranke für die Distanzen gefunden werden kann, wenn die Selbstähnlichkeit der Kurve einbezogen wird. Dieser Ansatz erlaubt jedoch auch keine weitere Auswertung nach dem Schema, das für die Lebesgue-Kurve angewendet werden kann. Dieselbe Problematik ergibt sich bei der Untersuchung der $\beta\Omega$-Indizierung, die sich mit ihren zwei Ableitungsregeln in der Analyse nochmals komplexer darstellt.

3.3.4 Experimentelle Ergebnisse

Die analytischen Ergebnisse in den vorangegangenen Abschnitten zeigen offensichtliche Lücken. Bei der Untersuchung des schlechtesten Falles ist eine scharfe Schranke für die Lebesgue-Kurve beweisbar. Im vorherigen Abschnitt sind untere Schranken für die Partitionierungsqualität der Hilbert- und der $\beta\Omega$-Indizierung gezeigt, jedoch keine entsprechenden oberen. Für den mittleren Fall konnte eine untere Schranke für die Lebesgue-Kurve angegeben werden, die jedoch schon aufgrund der wenigen Abschätzungen im Verlauf des Beweises als scharf angesehen werden kann.

Um die bekannten Schranken besser einordnen zu können, sind im Folgenden umfassende Simulationsergebnisse dargestellt. Im Gitter der Größe 1024×1024 werden alle möglichen Partitionen von einem gegebenen Volumen V untersucht und anschließend hinsichtlich des schlechtesten und mittleren Falles ausgewertet. An dieser Stelle ist auch die H-Indizierung in die Betrachtung mit aufgenommen, um deren schlechte Qualität im Bereich der Partitionierung von Gittern zu verdeutlichen.

Tabelle 3.1 fasst die experimentell ermittelten Partitionierungsqualitäten zusammen. Die fett gedruckten Einträge sind Ergebnisse, die nahe an bewiesenen Schranken liegen. Diese lassen vermuten, dass die gefunden Schranken recht scharf sind, obgleich der Beweis für die entsprechende obere Schranke fehlt.

Die Simulationsergebnisse sind in Abbildung 3.13 und 3.14 detailliert angegeben. Die Partitionierungsqualitäten des schlechtesten Falls sind stark oszillierend, so dass sie

$\mathcal{P}$	schlechtester Fall	mittlerer Fall
Hilbert	**1.87**	1.39
Lebesgue	**1.84**	**1.45**
$\beta\Omega$-Indizierung	**1.96**	1.37
H-Indizierung	2.3	1.55

Tabelle 3.1: Experimentell ermittelte Partitionierungsqualität der vier Indizierungsschemata

sich nicht übersichtlich in einem gemeinsamen Diagramm darstellen lassen. Bei allen Kurven, insbesondere jedoch bei der Lebesgue-Kurve, zeigt sich die Selbstähnlichkeit der raumfüllenden Kurven durch den sich regelmäßig wiederholenden Verlauf der Messkurven. Das Fehlen der hochfrequenten Oszillationen bei der Lebesgue-Kurve liegt an der Tatsache, dass der Rand der Partition der Größe eines umfassenden Rechtecks entspricht (vgl. Lemma 3.1). Damit wächst die Länge des Randes streng monoton. Der schneckenförmige Verlauf der anderen Kurven führt dagegen zu ständig schwankenden Partitionsrändern, da nicht jede Partitionsgröße den gleichen Anteil an schneckenförmigen Partitionsrändern ermöglicht. Da die schneckenförmigen Enden der Partitionen mit wachsendem Partitionsvolumen länger werden können, steigt bei diesen raumfüllenden Kurven der normierte Partitionsrand im untersuchten Bereich leicht an.

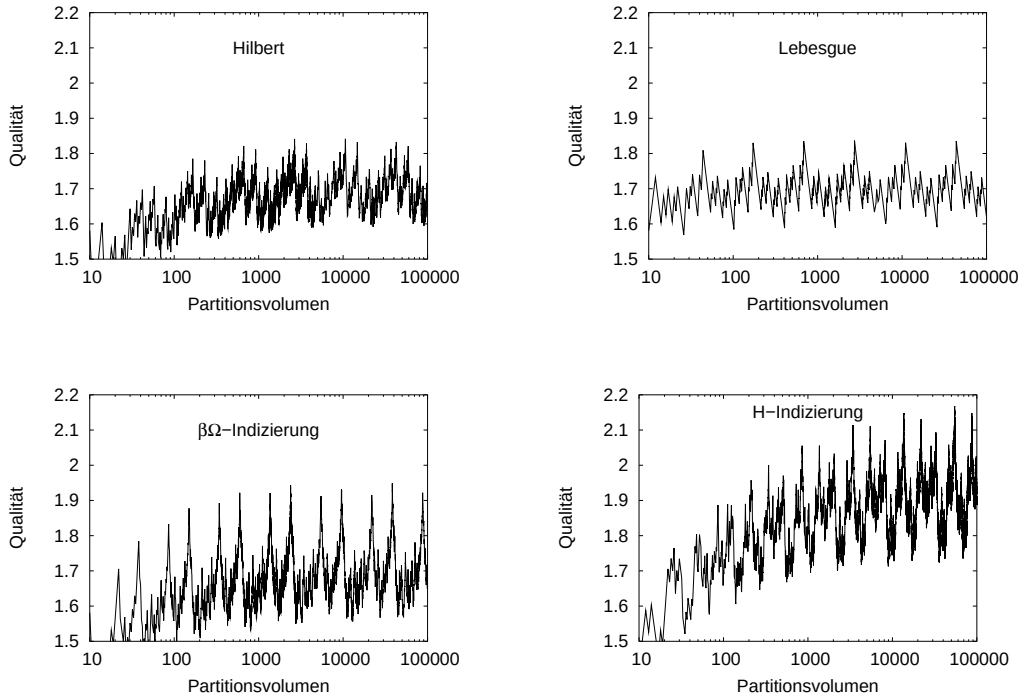


Abbildung 3.13: Partitionierungsqualität der vier Indizierungsschemata im schlechtesten Fall

Der Verlauf der Messwerte verdeutlicht auch, dass die Wahl der Definition der Partitionierungsqualität sinnvoll ist. Im untersuchten Bereich von Partitionsgrößen zwischen 10 und 100 000 Elementen ergeben sich abgesehen von den Oszillationen nahezu gleich bleibende Werte.

Der Unterschied in der Partitionierungsqualität von Lebesgue- und Hilbert-Kurve ist äußerst gering. Die Experimente lassen den Schluss zu, dass die gefundene Schranke von $\mathcal{P}_{\max}^{\text{Hilbert}} < 3\sqrt{\frac{5}{13}} \approx 1.86$ nahe der tatsächlichen Partitionierungsqualität im schlechtesten Fall liegt. Die $\beta\Omega$-Indizierung hat mit etwa 1.96 offensichtlich eine schlechtere Qualität. Diese hohen Werte werden jedoch nur an wenigen Stellen angenommen, so dass die Qualität der Partitionen bei vielen Volumina in der gleichen Größenordnung

liegt, wie bei der Hilbert- und der Lebesgue-Kurve.

Im Gegensatz zu den anderen Indizierungsschemata steigt bei der H-Indizierung der normierte Partitionsrand deutlich im untersuchten Raum. Die Grundstruktur einer besonders schlechten Partition ist ein gleichschenkliges Dreieck, dessen Schenkel an diagonalen Separatoren verlaufen. Diese Grundstruktur wird durch schneckenförmige Strukturen an den Endpunkten der dritten Kante ergänzt. Neben dem Qualitätsverlust durch längere schneckenförmige Strukturen wie auch bei der Hilbert- und der $\beta\Omega$-Kurve, wächst auch der normierte Rand des Dreiecks, der die Grundstruktur bildet. Daher ist in dem untersuchten Bereich die Konvergenz der Partitionierungsqualität für beliebig große Partitionen noch nicht deutlich. Für die Untersuchung bedeutend größerer Partitionsvolumina ist die entsprechende Vergrößerung des Basisgitters notwendig. Anderenfalls ist die Partition nicht beliebig im Gitter positionierbar und spiegelt damit nicht die Struktur und damit die Qualität im schlechtesten Fall wider. Die Wahl der Größe des Basisgitters stößt jedoch an die Grenze des Speicherraumes auf den Systemen, die für die Auswertung der Partitionierungseigenschaften der raumfüllenden Kurven zur Verfügung standen. Bei einer vorsichtigen Schätzung kann jedoch angenommen werden, dass die Partitionierungsqualität 2.3 nicht übersteigt.

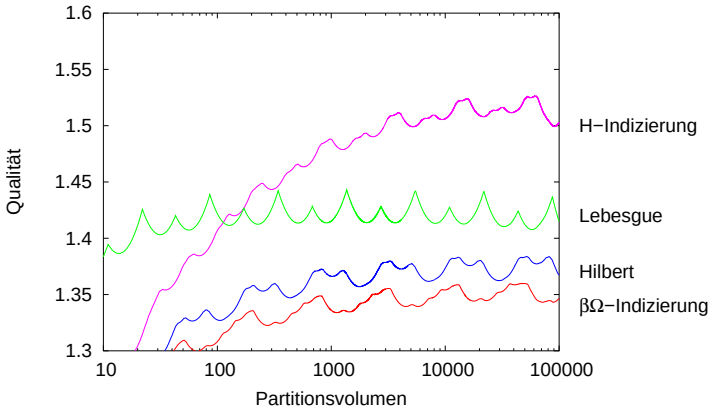


Abbildung 3.14: Partitionierungsqualität im mittleren Fall

In Abbildung 3.14 sind die Ergebnisse zum mittleren Fall dargestellt. Diese sind von großer Bedeutung, da sie eine Einordnung der einzigen analytischen Lösung, die der Lebesgue-Kurve, erlauben. Aufgrund der Mittelung der Partitionierungsqualitäten bei festem Volumen ergeben sich keine hochfrequenten Oszillationen im Verlauf der Messkurven. Andere strukturelle Eigenschaften bleiben erhalten: Die schneckenförmigen raumfüllenden Kurven zeigen einen Anstieg des normierten Randes, während die Lebesgue-Kurve eine gleich bleibende Qualität aufweist. Zudem konvergieren die Messwerte zur H-Indizierung weniger schnell, als die der anderen schneckenförmigen Kurven. Die Zyklen mit der Länge vier geben die Selbstähnlichkeit der Kurven und damit auch die der induzierten Partitionen wieder. Der stückweise parabelförmige Verlauf der Messwerte der Lebesgue-Kurve ergibt sich aus den stückweise linear wachsenden Partitionsrändern zusammen mit der vorgenommenen Normierung und der logarithmischen

Skalierung der x-Achse (vgl. Beweis zu Lemma 3.2).

Die Rangfolge bei der mittleren Partitionierungsqualität zeigt die $\beta\Omega$-Indizierung vor Hilbert- und Lebesgue-Kurve. Wie zu erwarten, zeigt die H-Indizierung auch im mittleren Fall die schlechtesten Partitionierungseigenschaften. Damit ergibt sich für die drei rekursiven raumfüllenden Kurven die umgekehrte Ordnung wie bei der Untersuchung des schlechtesten Falls:

$$\mathcal{P}_{\max}^{\text{Lebesgue}} < \mathcal{P}_{\max}^{\text{Hilbert}} < \mathcal{P}_{\max}^{\beta\Omega-\text{Ind}} < \mathcal{P}_{\max}^{\text{H}-\text{Ind}} \qquad \text{und}$$

$$\mathcal{P}_{\text{avg}}^{\beta\Omega-\text{Ind}} < \mathcal{P}_{\text{avg}}^{\text{Hilbert}} < \mathcal{P}_{\text{avg}}^{\text{Lebesgue}} < \mathcal{P}_{\text{avg}}^{\text{H}-\text{Ind}} \quad .$$

3.4 Partitionierungsqualität in unstrukturierten Netzen

Der vorangegangene Abschnitt hat sich bei der Untersuchung der Qualität von Partitionen auf Basis raumfüllender Kurven auf die Graphstruktur des Gitters beschränkt. Anwendungen im Bereich der FEM basieren jedoch im Allgemeinen auf unstrukturierten Netzen, deren Struktur sich deutlich vom Gitter unterscheiden kann. Die oben gezeigten guten Eigenschaften der raumfüllenden Kurven bei der Partitionierung von Gittern ergibt sich in weiten Teilen aus der Tatsache, dass die Kurven über die gleiche Struktur definiert sind. Zudem besteht aufgrund der regelmäßigen Struktur des Gitters kein großer Unterschied zwischen einer geometrisch oder einer strukturell basierten Partitionierungsmethode. Da sich die Ergebnisse nicht übertragen lassen, befasst sich dieser Teil der Arbeit mit der Frage, wie gut sich die Kurven zur Partitionierung allgemeiner FE-Netze einsetzen lassen. Es zeigt sich, dass die Qualität der Partitionierungen keinesfalls so gut ist wie im Gitter. Der Verlust gegenüber anderen Partitionierungsverfahren ist jedoch in den meisten Fällen begrenzt. Um eine Vergleichbarkeit mit anderen Partitionierungsverfahren zu ermöglichen, wird an dieser Stelle die Knotenpartitionierung betrachtet.

Im folgenden Abschnitt werden die charakteristischen Eigenschaften der Partitionierung mittels Indizierung an einem Beispiel aufgezeigt. Danach wird die Partitionierungsqualität an verschiedenen zwei- und dreidimensionalen Netzen untersucht. Die Ergebnisse werden mit den Resultaten des Partitionierungswerkzeugs Metis verglichen. Da für große irreguläre Graphen im Allgemeinen keine optimalen Lösungen für die k-Partitionierung existieren, kann ein Vergleich nur mit einer anderen heuristischen Methode durchgeführt werden. Die Qualität von Metis ist in verschiedenen Publikationen gezeigt worden, so dass sich Metis als hochwertiger Vergleich eignet [35, 45, 73]. Metis beinhaltet zwei Varianten der Partitionierung: Die direkte k-Partitionierung, die im Folgenden als kMetis bezeichnet wird und die k-Partitionierung durch rekursive Bisektion, pMetis.

3.4.1 Beispiel: 5-Partitionierung des Benchmarks *biplane*

In Abschnitt 3.2.2 sind die prinzipiellen Stärken und Schwächen der Partitionierung mittels raumfüllender Kurven am Beispiel der 5-Partitionierung des 16×16-Gitters diskutiert. Im Folgenden werden die untersuchten Kurven hinsichtlich ihrer Partitionierungsqualität in einem unstrukturierten Netz verglichen.

Abbildung 3.15 zeigt ein zweidimensionales FE-Netz mit zwei Löchern. Die Gitterstruktur ist in einigen Bereichen, vornehmlich an den Rändern der Löcher, mehrstufig mittels Viertelung der Gitterzellen verfeinert. Rund 90 Prozent aller Knoten und Kanten des Netzes liegen im Gebiet der höchsten Verfeinerung, so dass der resultierende Kantenschnitt vom Verlauf der Partitionsgrenzen in diesem Bereich dominiert wird. Der rechte Teil von Abbildung 3.15 zeigt die von *k*Metis berechnete 5-Partitionierung. Der Kantenschnitt beträgt 299. Die Partitionsränder liegen im Bereich der Löcher des Netzes und verlaufen im Netz über relativ kurze Wege durch die Gebiete der höchsten Verfeinerung. Lediglich zwischen der blauen und der türkisen Partition verläuft die Grenze im Bereich der höchsten Verfeinerung. Dort ergibt sich etwa ein Drittel (98) des gesamten Kantenschnittes.

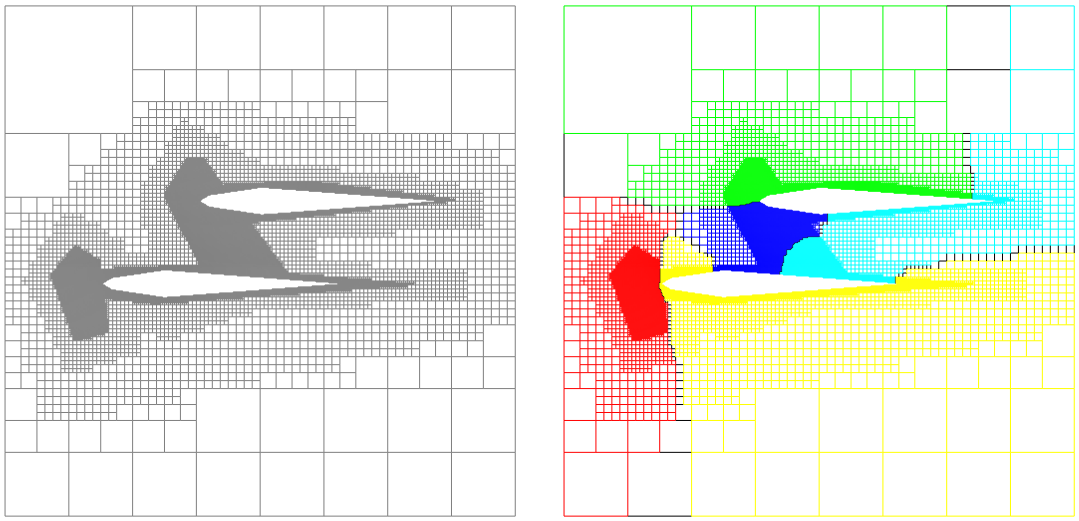


Abbildung 3.15: Benchmark *biplane* (links) und 5-Partitionierung durch *k*Metis (rechts)

Die 5-Partitionierungen, die sich durch die Anwendung der untersuchten raumfüllenden Kurven ergeben, sind Abbildung 3.16 angegeben. Diese geometrischen Verfahren vernachlässigen die Struktur des Graphen, so dass Partitionsgrenzen nicht entlang der Löcher verlaufen und auch vermehrt im Bereich der höchsten Netzverfeinerung liegen. Für die vier Indizierungsschemata ergeben sich Kantenschnitte von 627 (Hilbert), 611 (Lebesgue), 863 (Sierpiński) und 615 ($\beta\Omega$). Damit liegt der Kantenschnitt um einen Faktor zwei bis drei über dem der Partitionierungsheuristik. Dieser Qualitätsverlust ist deutlich größer als in regelmäßigen Gittern.

Bei allen vier Partitionierungen ergeben sich aufgrund der Löcher im Netz ein oder

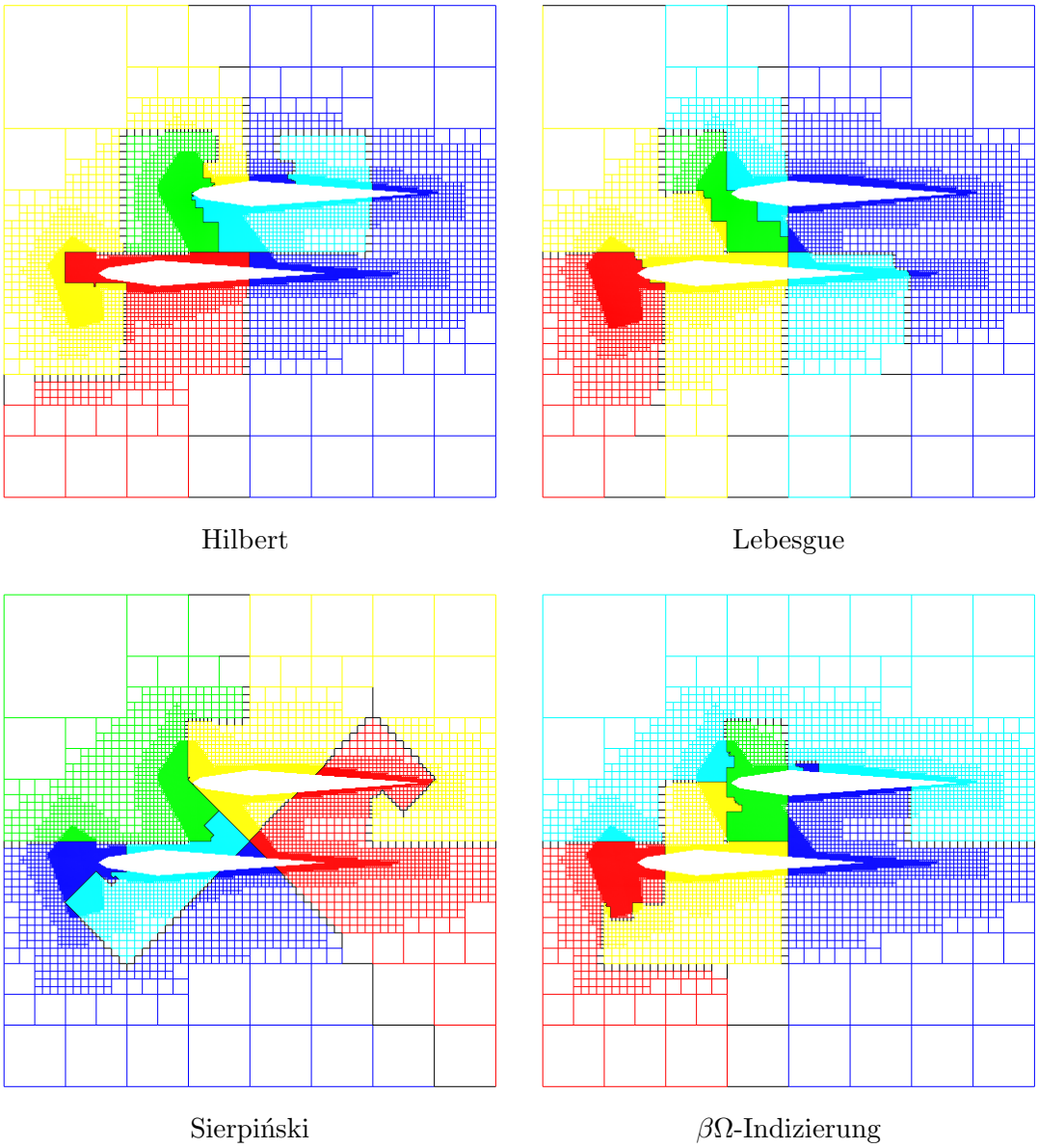


Abbildung 3.16: *biplane*: 5-Partitionierungen über raumfüllende Kurven

zwei nicht-zusammenhängende Partitionen. Desweiteren sind einige Partitionen nur schwach zusammenhängend, d. h. sie zerfallen bei der Entfernung weniger Kanten. Diese Struktur impliziert eine relativ große Anzahl externer Kanten. Abgesehen von diesen offensichtlichen Schwächen führen die Indizierungsschemata Hilbert, Lebesgue und $\beta\Omega$, die lediglich auf vertikalen und horizontalen Schnitten basieren, zumeist zu stückweise geradlinigen, achsenparallelen Partitionsgrenzen mit geringem Kantenschnitt in einem solchen gitterähnlichen Netz. Dahingegen führen die vornehmlich diagonalen Separationslinien der Sierpiński-Kurve zu einem deutlich höheren Kantenschnitt trotz vergleichbar kompakter Partitionen. Die gewundene Struktur der Kurven Hilbert, Sierpiński und $\beta\Omega$ zeigen sich auch in diesem unstrukturierten Netz an den schneckenförmigen Enden

der Partitionen. Dahingegen ergeben sich bei der Lebesgue-Kurve treppenförmige Partitionsgrenzen, z. B. zwischen der gelben und der grünen Partition.

Der Kantenschnitt bei der 5-Partitionierung des gegebenen Netzes bei einer zeilenweisen Indizierung der Knoten liegt bei 798 und damit im Bereich der Qualität der raumfüllenden Kurven. Es ergeben sich lediglich geradlinige Schnitte und die Löcher bilden Teile der Partitionsgrenzen. Die weiteren Untersuchungen zeigen jedoch, dass bei einer Partitionierung in mehr Gebiete die Vorteile der kompakten Struktur, die sich aus den raumfüllenden Kurven ergeben, größer sind, als die Nachteile der schnecken- oder treppenförmigen Struktur an deren Gebietsränder.

3.4.2 Beschreibung der Netze

Die experimentellen Untersuchungen finden auf Basis einer Sammlung von Netzen statt, die schon in anderen Veröffentlichungen als Benchmarks eingesetzt wurden [7, 21, 27, 37, 45, 68, 79]. Sie umfassen sowohl zwei- als auch dreidimensionale unstrukturierte Netze. Da für den Einsatz der raumfüllenden Kurven Koordinaten der Knoten gegeben sein müssen, schränkt sich die Auswahl möglicher Benchmark-Netze ein. Abgesehen von dem regulären Gitter entstammen alle Netze FE-Anwendungen.

Die wichtigsten charakteristischen Eigenschaften der untersuchten zweidimensionalen Netze sind in Tabelle 3.2 zusammengefasst. *grid* ist ein zweidimensionales Gitter der Größe 100×100. Es dient dem Vergleich dieser experimentellen Untersuchung mit den ermittelten theoretischen Ergebnissen zur Partitionierungsqualität im vorangegangenen Abschnitt. Alle anderen Netze haben eine irreguläre Struktur mit erhöhter Netzdichte in einzelnen Bereichen. Die Netze *airfoil*, *crack* und *big* sind triangulierte Gebiete mit einem hohen Knotengrad. Dabei ist *crack* durch die reguläre Verfeinerung eines Gitters entstanden, so dass alle Kanten entweder achsenparallel oder diagonal verlaufen. Die anderen drei Netze haben eine gitterähnliche Struktur mit lediglich achsenparallel verlaufenden Kanten. Die Netze *airfoil*, *big* und *biplane* haben Löcher im diskretisierten Gebiet.

Netz	$\lvert V\rvert$	$\lvert E\rvert$	$\deg_{\min}$	$\deg_{\text{avg}}$	$\deg_{\max}$
grid	10 000	19 600	2	3.96	4
airfoil	4 253	12 289	3	5.78	9
crack	10 240	30 380	3	5.93	8
big	15 606	45 878	3	5.88	10
biplane	21 701	42 038	2	3.87	4
stufe	24 010	46 414	2	3.87	4
shock	36 476	71 290	2	3.91	4

Tabelle 3.2: Zweidimensionale Benchmarks

Netz	$\|V\|$	$\|E\|$	$\deg_{\min}$	$\deg_{\text{avg}}$	$\deg_{\max}$
pwt	36 519	144 794	1	7.93	15
brack	62 631	366 559	3	11.71	32
rotor	99 617	662 431	5	13.30	125
wave	156 317	1 059 331	3	13.55	44
hermes	320 194	3 722 641	4	23.25	56

Tabelle 3.3: Dreidimensionale Benchmarks

Tabelle 3.3 zeigt die charakteristischen Eigenschaften der dreidimensionalen Benchmarks. Alle Netze sind durch Tetraeder diskretisierte Objekte oder Gebiete. Das Netz *pwt* dient der Simulation in einem kleinen Rohrsystem. Die Oberfläche ist in Abbildung 3.6 (S. 32) dargestellt. *brack* ist die Diskretisierung eines mechanischen Bauteils. Die Beispiele *rotor*, *wave* und *hermes* sind Netze zur Strömungssimulation. Im Gegensatz zu den vorhergehenden umfassen sie ein konvexes Gebiet mit einem darin enthaltenen Objekt. *rotor* dient der Simulation an einer Rotorspitze, welche von einem flachen zylindrischen Simulationsraum umgeben ist. Von der Rotorspitze im Zentrum des Zylinders verläuft eine kegelförmiges Gebiet mit einer sehr feinen Diskretisierung zum Rand des Zylinders. *wave* und *hermes* sind quaderförmige Bereiche die eine Flugzeughälfte bzw. die Spitze eines Raumgleiters beinhalten. Die Netze sind im Bereich der umströmten Objekte verfeinert.

3.4.3 Ergebnisse

3.4.3.1 Zweidimensionale Netze

Der Benchmark *grid* dient zur Bewertung der analytischen Resultate aus Abschnitt 3.3. Abbildung 3.17 vergleicht den Kantenschnitt der Partitionierung mittels der untersuchten raumfüllenden Kurven mit dem von *k*Metis. Der Kantenschnitt der Partitionierung wächst mit der Anzahl der Partitionen für alle Heuristiken. Abgesehen von kleineren Zweierpotenzen liefert *k*Metis die Partitionierung mit dem kleinsten Kantenschnitt. Für die 2-, 4-, 8- und 16-Partitionierung ist eine Zerlegung des 100×100 Gitters in Quadrate bzw. Rechtecke möglich, die von den raumfüllenden Kurven Hilbert, Lebesgue und $\beta\Omega$ gefunden werden. Die Differenz im Kantenschnitt zwischen den Partitionierungen auf Basis der raumfüllenden Kurven und der Lösung von *k*Metis wächst mit steigender Anzahl an Partitionen, während der relative Qualitätsgewinn fällt.

In Abbildung 3.18 ist der mit den raumfüllenden Kurven erzielte Kantenschnitt in Relation zu der von *k*Metis gelieferten Lösung dargestellt. An den Partitionierungen, die einer Zweierpotenz entsprechen, spiegelt sich ein wesentlicher Unterschied bei der Konstruktion der Kurven am deutlichsten wider. Während Hilbert-, Lebesgue- und $\beta\Omega$-Kurve lediglich über horizontale und vertikale Separatoren definiert sind, besteht die

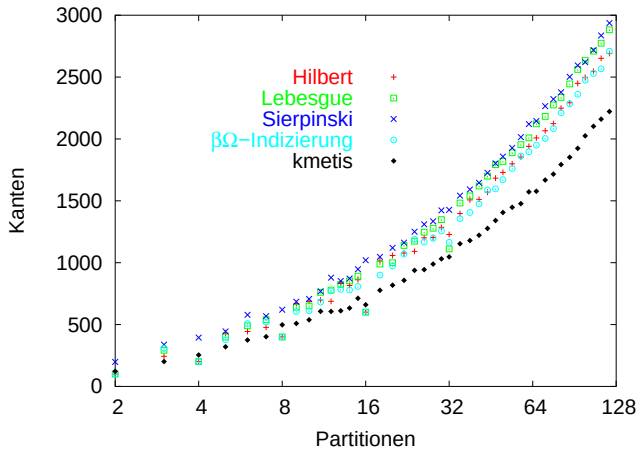


Abbildung 3.17: Kantenschnitt im Beispiel *grid*

Konstruktion der Sierpiński-Kurve zusätzlich aus diagonalen Separatoren. Erstgenannte Kurven finden eine Zerlegung in Quadrate bzw. Rechtecke, während die Partitionen der Sierpiński-Kurve aus Dreiecken bestehen. Für die 2-, 4-, 8- und 16-Partitionierung sind die Kantenschnitte der rekursiven raumfüllenden Kurven denen von *k*Metis um bis zu 17 Prozent überlegen. Dagegen sind die Partitionierungen, die auf der Sierpiński-Kurve basieren um etwa 50 bis 80 Prozent schlechter. Für größere Zweierpotenzen ist keine triviale Zerlegung in Rechtecke möglich und die Kurven liefern schlechtere Lösungen als *k*Metis. Dies gilt auch für beliebige andere Anzahlen an Partitionen.

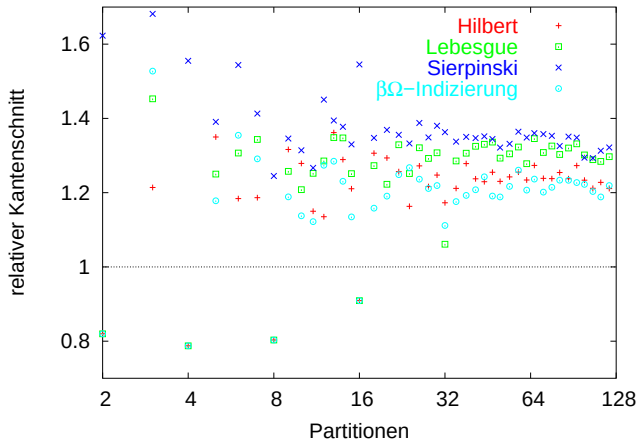


Abbildung 3.18: Partitionierungsqualität relativ zu *k*Metis im Benchmark *grid*

Für alle untersuchten Partitionierungen zeigt sich, dass die Sierpiński-Kurve die schlechtesten Resultate liefert. Etwas besser verhält sich die Lebesgue-Kurve, deren Vorsprung jedoch mit wachsender Partitionsanzahl schrumpft. Am besten verhalten sich Hilbert- und $\beta\Omega$-Kurve. Letztere weist ab der 16-Partitionierung einen Qualitätsverlust von weniger als 30 Prozent gegenüber *k*Metis auf. Diese Reihenfolge entspricht den Ergebnissen aus Abschnitt 3.3.4. Dort wird die mittlere Qualität bzgl. unterschiedlicher Partitionsvolumina verglichen, während hier die Partitionsanzahl skaliert wird. Eine

direkte Umrechnung von der Partitionsanzahl auf die Partitionsgröße ist nur eingeschränkt möglich, da die Untersuchungen auf Gittern von deutlich unterschiedlicher Größe basieren. Außerdem sind die Experimente in Abschnitt 3.3.4 als Näherung für unendlich große Gitter eingesetzt. Dort wird der gesamte Rand der Partitionen summiert, d. h. auch der Teil, der am Rand des Gitters liegt. Die Messungen zu dem hier untersuchten Kantenschnitt liefern jedoch keinen Wert für den Rand des Gitters.

In Abbildung 3.19 sind die Kantenschnitte relativ zur Wurzel der Partitionsvolumina dargestellt.

$$\text{normierter Kantenschnitt} = \frac{\text{Kantenschnitt} \cdot 2}{\#\text{Partitionen} \cdot \sqrt{\#\text{Knoten je Partition}}} \tag{3.31}$$

Die Normierung erlaubt die Darstellung der Partitionierungsqualität für die untersuchten raumfüllenden Kurven gemeinsam mit dem Vergleichswert von kMetis. Sie ergibt sich aus der Erwartung, dass die Oberfläche und damit auch die Anzahl der Schnittkanten einer Partition im zweidimensionalen Raum proportional zur Wurzel des Volumens wächst und ist an die Definition der Partitionierungsqualität im Gitter angelehnt (Def. 3.1, S. 34). Lediglich die Konstante 4 für die Oberfläche eines Quadrates fehlt. Die Normierung dient dem übersichtlicheren qualitativen Vergleich der Partitionierungen von Netzen unterschiedlicher Größe und mit unterschiedlicher Partitionsanzahl. Der normierte Wert wird für eine große Anzahl an Partitionen gegen den mittleren Grad des Graphen konvergieren. Der Verlauf der Messwerte kann einen Aufschluss über die Qualität der Partitionierungen geben: Liegt der normierte Kantenschnitt im gesamten Bereich unter dem mittleren Kantenschnitt, sind die Partitionen von höherer Qualität, im anderen Falle von schlechterer Qualität. Die Werte zeigen jedoch nicht, ob eine mögliche gute Qualität an der Stärke des Partitionierungsverfahrens oder an den guten Eigenschaften des Netzes liegt.

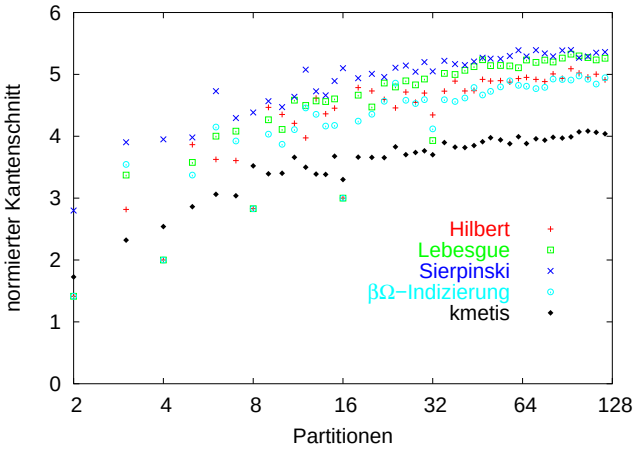


Abbildung 3.19: Partitionierungsqualität am Beispiel *grid*

Der normierte Kantenschnitt wächst für alle Partitionierungsstrategien zunächst mit der Anzahl an Partitionen. Dieser Effekt ergibt sich aus der Tatsache, dass sich das

Verhältnis vom äußeren Rand zu den inneren Rändern zwischen den Partitionen verringert. Unter Einbeziehung von 400 virtuellen äußeren Kanten ergibt sich ein nahezu horizontaler Verlauf der Messpunkte, die im Falle von kMetis zumeist im Bereich zwischen 4.5 und 5 liegen. Diese Werte zeigen, dass kMetis für dieses Beispiel i. A. Partitionierungen liefert, die höchstens 20 Prozent schlechter sind als eine optimale Partitionierung. Unter Einbeziehung virtueller äußerer Kanten wird der normierte Kantenschnitt einer optimalen Partitionierung vier nicht unterschreiten. Im Falle einer weiteren Erhöhung der Anzahl der Partitionen tendiert der normierte Kantenschnitt offensichtlich gegen vier, dem mittleren Knotengrad im zweidimensionalen Gitter. Die Untersuchungen an verschiedenen Beispiel-Netzen haben gezeigt, dass die leistungsfähigen Partitionierungsheuristiken kMetis und pMetis für eine beliebige Anzahl an Partitionen i. A. normierte Kantenschnitte unter bzw. nur leicht über dem mittleren Knotengrad liefern. Der Hochpunkt der Messwertkurve liegt häufig im Bereich zwischen $|V|^{\frac{1}{2}}$ und $|V|^{\frac{3}{4}}$ Partitionen für die hier untersuchten Partitionierungsverfahren.

Die Messwerte zu den Partitionierungseigenschaften der raumfüllenden Kurven im Gitter erlauben einen Vergleich mit dem analytischen Ergebnis in Abschnitt 3.3.3 zur Lebesgue-Kurve und den experimentellen Untersuchungen in Abschnitt 3.3.4 zu der Partitionierungsqualität der Kurven im mittleren Fall. Für diesen Vergleich müssen wiederum 400 virtuelle äußere Kanten mit einbezogen werden. Der normierte Kantenschnitt der Hilbert- und der $\beta\Omega$-Kurve erhöht sich auf 5.2 bis 5.6, welcher einer Partitionierungsqualität von 1.3 bis 1.4 entspricht. Für die Lebesgue-Kurve ergeben sich Werte von etwa 5.7 (normierter Kantenschnitt) bzw. 1.425 (Partitionierungsqualität). Bei der Addition von 400 virtuellen Kanten hat die Sierpiński-Kurve bei geringen Partitionsanzahlen den höchsten normierten Kantenschnitt von 6.4. Sie zeigt das gleiche Verhalten wie die mit ihr verwandte H-Indizierung, die bei großen Partitionsvolumen die schlechteste Partitionierungsqualität aufweist (vgl. Abb. 3.14). Dagegen haben Sierpiński- und Lebesgue-Kurve bei der 128-Partitionierung einen vergleichbaren normierten Kantenschnitt, was der Beobachtung aus Abschnitt 3.3.4 entspricht, dass H-Indizierung und Lebesgue-Kurve bei einer Partitionsgröße von etwa 100 eine ähnlich hohe Partitionierungsqualität haben.

Abbildung 3.20 zeigt den normierten Kantenschnitt für das Beispiel *biplane* dessen 5-Partitionierung in Abschnitt 3.4.1 analysiert wurde (vgl. Abb. 3.15). Aufgrund seiner gitterähnlichen Struktur ergibt sich ein zum Gitter vergleichbarer Verlauf der Messwerte. In diesem Beispiel lässt sich kein genereller Qualitätsunterschied zwischen den rekursiven raumfüllenden Kurven erkennen. Der normierte Kantenschnitt steigt bis auf 5 bei 128 Partitionen, einem zum Gitter vergleichbaren Wert. Die Messkurve von kMetis verläuft deutlich flacher und steigt auf lediglich 3.5 für die 128-Partitionierung. Bei ähnlichem Knotengrad ist es ein Indiz dafür, dass dieses Beispiel besser zu partitionieren ist. Wenn möglichst viele Partitionsgrenzen im Bereich der Löcher verlaufen, ergibt sich ein deutlich geringerer Kantenschnitt als im regelmäßigen Gitter.

Die geometrischen Verfahren liefern einen ähnlich großen normierten Kantenschnitt wie im Beispiel *grid*. Da sie die Struktur des Graphen nicht berücksichtigen, ist ein geringerer Kantenschnitt aufgrund der Löcher nur zufällig möglich. Die rekursiven Varianten

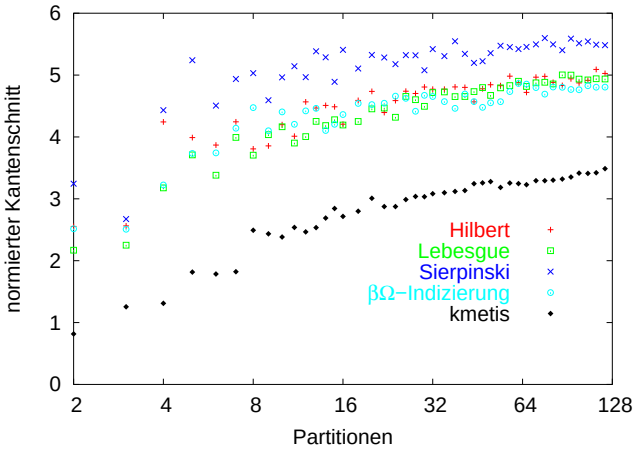


Abbildung 3.20: Partitionierungsqualität am Beispiel *biplane*

liefern leicht bessere, die Sierpiński-Kurve leicht schlechtere Werte, so dass der relative Unterschied deutlicher wird. Zudem lässt sich zwischen den Kurven Hilbert, Lebesgue und $\beta\Omega$ keine eindeutige Ordnung bzgl. ihrer Qualität erkennen.

Abbildung 3.21 zeigt den normierten Kantenschnitt für das Netz *big*, der zu der Gruppe der triangulierten Gebiete zählt. Aufgrund eines mittleren Knotengrades von 5.88 liegt der normierte Kantenschnitt für alle Verfahren höher. Bei den Messwerten von kMetis ergibt sich wiederum ein flacherer Anstieg, welcher mit den vorhandenen Löchern im Netz zu begründen ist. Dahingegen liegt der normierte Kantenschnitt der raumfüllenden Kurven innerhalb eines schmalen Intervalls, wenn man von den groben Partitionierungen absieht. Dieser Effekt erklärt sich zum einen aus der geometrischen Heuristik, welche die Löcher im Netz nicht nutzt. Zum anderen ist der Gebietsrand mit 27 Knoten sehr grob vernetzt, so dass sich an dieser Stelle kaum ein Einfluss auf den normierten Kantenschnitt ergibt, wie er für das Beispiel *grid* festgestellt wurde.

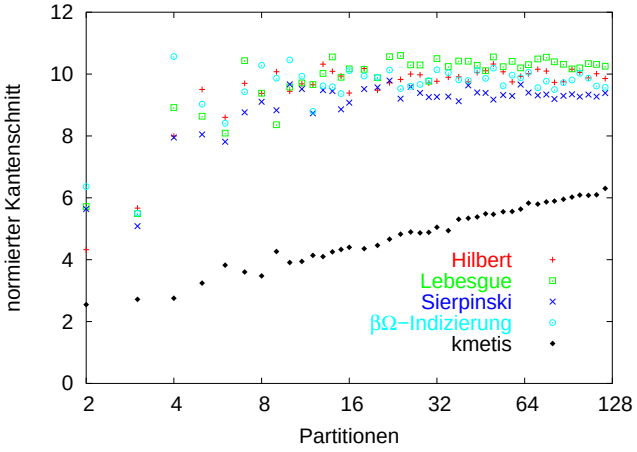


Abbildung 3.21: Partitionierungsqualität am Beispiel *big*

Die Ordnung der raumfüllenden Kurven hat sich jedoch bzgl. ihrer Partitionierungsqualität geändert. Die Sierpiński-Kurve führt in fast allen Fällen zu einer Zerlegung mit

geringerem Kantenschnitt. Da die Kanten nicht achsenorientiert verlaufen, sondern beliebige Lagen in der Ebene einnehmen, ist die Orientierung der Separationslinien der einzelnen Verfahren für dieses Beispiel nicht von Bedeutung. Die in Abschnitt 3.2.2 anhand der 5-Partitionierung des 16×16-Gitters beobachtete kompakte Struktur mit nur kurzen schneckenförmigen Endungen kommt in diesem Beispiel zum tragen. Die Lebesgue-Kurve liefert für dieses Netz die schlechtesten Resultate, wobei der Unterschied zu den beiden anderen rekursiven raumfüllenden Kurven äußerst gering ist.

Der relative Qualitätsunterschied der raumfüllenden Kurven gegenüber *k*Metis ist in diesem Beispiel deutlich größer als im Gitter und im Netz *biplane*, insbesondere bei kleiner Partitionsanzahl. Abbildung 3.22 zeigt die relative Qualität für die unstrukturierten zweidimensionalen Benchmarks im Überblick. Für die triangulierten Netze *airfoil*, *crack* und *big* sind die Messwerte auf Basis der Sierpiński-Kurve, bei den anderen Netzen die der Lebesgue-Kurve, als Stellvertreter der rekursiven raumfüllenden Kurven, ausgewählt, da diese die jeweils besten Resultate liefern. Es zeigt sich, dass die Streuung der Messwerte mit wachsender Anzahl an Partitionen abnimmt. Für die gitterähnlichen Netze liegt die relative Verschlechterung in den meisten Fällen unter 2, bei mehr als 16 Partitionen unter 1.5. Bei dem Netz *crack* ist der Anstieg an Schnittkanten gegenüber *k*Metis lediglich bei geringer Anzahl an Partitionen höher. Dieses Netz ist trianguliert, jedoch aus einer lokal verfeinerten Gitterstruktur entstanden. Seine Grundstruktur ähnelt damit der Konstruktion der Indizierungsschemata. Bei den beiden Netzen ohne jede regelmäßige Struktur führen die raumfüllenden Kurven (in diesem Falle die Sierpiński-Kurve) zu sichtbar schlechteren Ergebnissen. Bei dem Netzen *airfoil* liegt der relative Kantenschnitt erst ab 24 Partitionen unter 1.5, bei dem Netz *big* ist diese Schranke erst bei über 128 Partitionen unterschritten.

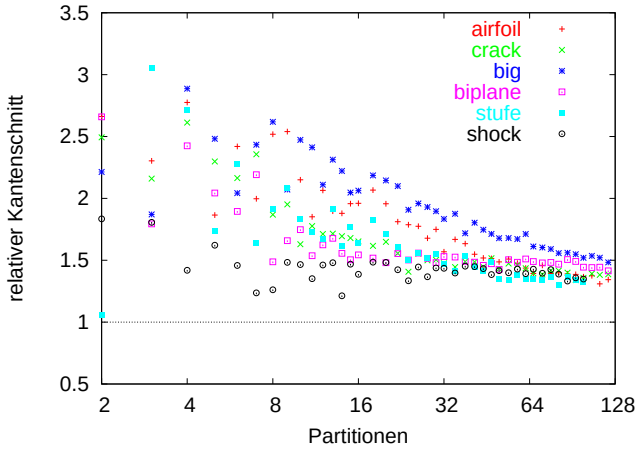


Abbildung 3.22: Qualität der Partitionierung relativ zu *k*Metis: zweidimensionale Netze

Die Tabellen 3.4 und 3.5 geben eine Übersicht über den absoluten Kantenschnitt der 16- bzw. 64-Partitionierungen der zweidimensionalen Netze durch Metis und durch die raumfüllenden Kurven. Das beste Ergebnis, das mit einer der raumfüllenden Kurve erzielt wird, ist jeweils hervorgehoben. An den Messwerten zeigt sich erneut der Einfluss der Konstruktionsvorschrift der raumfüllenden Kurven auf den erzielten Kantenschnitt.

Netz	kMetis	pMetis	Hilbert	Lebesgue	Sierpiński	$\beta\Omega$-Ind.
grid	660	706	**600**	**600**	1020	**600**
airfoil	555	574	1204	1215	**1125**	1276
crack	1218	1255	2060	2108	**2013**	2183
big	1099	1056	2345	2540	**2267**	2526
biplane	800	812	1253	**1235**	1593	1285
stufe	759	723	1251	**1245**	1503	1389
shock	1208	1233	1837	**1675**	2050	1736

Tabelle 3.4: Kantenschnitt der 16-Partitionierungen der zweidimensionalen Benchmarks

Netz	kMetis	pMetis	Hilbert	Lebesgue	Sierpiński	$\beta\Omega$-Ind.
grid	1543	1599	**1699**	1713	2141	1748
airfoil	1528	1572	2430	2501	**2265**	2408
crack	2781	2847	4031	4271	**3927**	4084
big	2843	2953	5025	5198	**4854**	4919
biplane	1906	2023	2867	2888	3229	**2844**
stufe	2268	2303	3131	**3062**	3594	3087
shock	2902	2889	3908	3915	4355	**3761**

Tabelle 3.5: Kantenschnitt der 64-Partitionierungen der zweidimensionalen Benchmarks

Die rekursiven Verfahren liefern in den meisten Fällen Resultate, die in einem schmalen Intervall liegen, während das Ergebnis der Sierpiński-Kurve deutlich abweicht. Welche der beiden Gruppen von raumfüllenden Kurven das bessere Ergebnis liefert, ist von der Struktur des Netzes abhängig. Im Falle des Gitters oder einer gitterähnlichen Struktur, wie *biplane*, *stufe* und *shock*, zeigen die rekursiven raumfüllenden Kurven das beste Ergebnis. Enthält das Netz viele diagonale Kanten oder solche mit beliebiger Richtung erzielt die Sierpiński-Kurve das beste Ergebnis.

Innerhalb der rekursiven raumfüllenden Kurven lässt sich keine eindeutige Ordnung über deren Qualität finden. Bei den 16-Partitionierungen scheint die Lebesgue-Kurve die besten Ergebnisse zu erzielen, jedoch ist die Menge der untersuchten Netze und der Unterschied zu den Resultaten der anderen Kurven zu klein, als dass eine generelle Tendenz festgestellt werden könnte. Die Hilbert-Kurve führt bei den 16-Partitionierungen für die drei triangulierten Netze zu den zweitbesten Ergebnissen hinter der Sierpiński-Kurve. Beide Beobachtungen lassen sich bei der Untersuchung der 64-Partitionierungen nicht feststellen.

3.4.3.2 Dreidimensionale Netze

Abbildung 3.23 zeigt den Verlauf des normierten Kantenschnitts für den dreidimensionalen Benchmark *hermes*. Im Falle des dreidimensionalen Raumes ist die Normierung des Kantenschnittes entsprechend der erwarteten Oberfläche von Körpern im Raum

angepasst:

$$\text{normierter Kantenschnitt} = \frac{\text{Kantenschnitt} \cdot 2}{\#\text{Partitionen} \cdot \sqrt[3]{\#\text{Knoten je Partition}^2}} \tag{3.32}$$

Aufgrund des hohen mittleren Kantengrades ergibt sich ein hoher normierter Kantenschnitt. Der qualitative Verlauf der Messwerte ist mit dem der zweidimensionalen vergleichbar. Bei kMetis zeigt sich ein zunächst deutlich flacheren Anstieg als bei beiden raumfüllenden Kurven. Mit größer werdender Partitionsanzahl nimmt der relative Abstand zu kMetis ab.

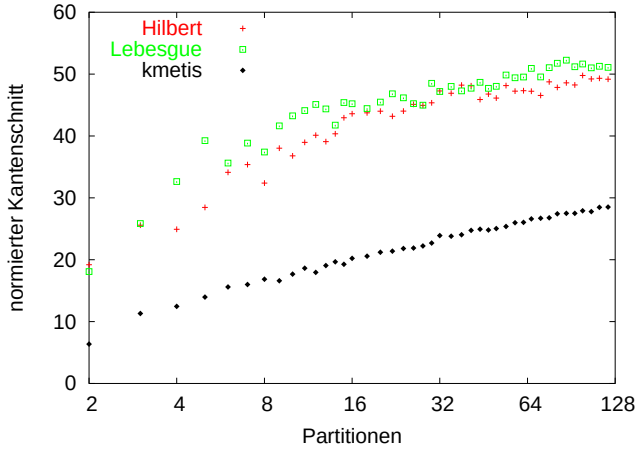


Abbildung 3.23: Partitionierungsqualität am Beispiel *hermes*

Abbildung 3.24 zeigt die Qualität der Partitionierungen beim Einsatz der Hilbert-Kurve für die untersuchten dreidimensionalen Benchmarks. Der Kantenschnitt liegt im Einzelfall um den Faktor 8.7 über dem der von kMetis ermittelten Partitionierung. Abgesehen vom Netz *rotor* liegt der relative Kantenschnitt in fast allen Fällen unter 3. Bei steigender Anzahl an Partitionen fällt dieser und liegt bei über 16 Partitionen unter 2.5, in den meisten Fällen unter 2. Der relative Kantenschnitt liegt deutlich höher als bei den untersuchten zweidimensionalen Netzen.

Zum Einen liegt dies an der Struktur der Netze, es sind alle tetraedierte Gebiete. Bei den zweidimensionalen Benchmarks hat sich gezeigt, dass die rekursiven Kurven sich bei triangulierten Netzen schlechter verhalten als bei gitterähnlichen Netzen. Das gleiche Phänomen ergibt sich im dreidimensionalen Raum. Es ist bisher jedoch keine dreidimensionale Erweiterung der Sierpiński-Kurve bekannt, die diese Netze in geeigneter Form indizieren kann.

Zum Anderen haben die Strukturen von verfeinerten Gebieten und Löchern einen stärkeren Einfluss auf die Qualität der Partitionierung, da sie mehr Varianten bzgl. ihrer Dimensionalität ermöglichen. Die Netze selbst, aber auch die speziellen Gebiete, wie Verfeinerungen oder Löcher, haben oft eine lediglich 1- oder 2-dimensionale Ausdehnung. Diese sind bei der Partitionierung mit minimalem Schnitt von großer Bedeutung, werden aber bei der Indizierung über raumfüllende Kurven nicht beachtet.

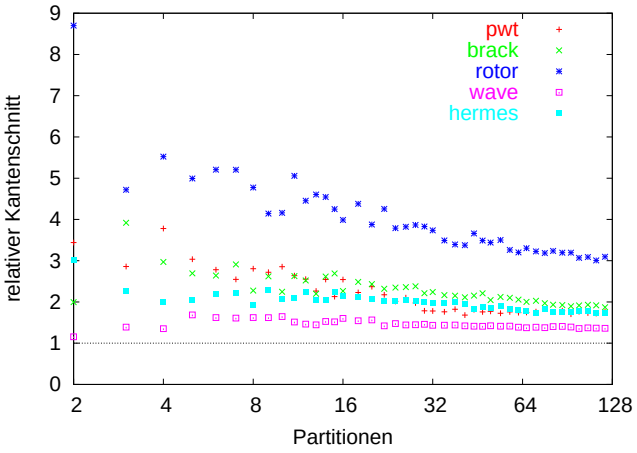


Abbildung 3.24: Qualität der Partitionierung relativ zu kMetis: dreidimensionale Benchmarks

Die Netze *hermes* und *wave* sind diskretisierte quaderförmige Gebiete mit einem enthaltenen Objekt, dessen Umströmung simuliert werden soll. Die Partitionierungseigenschaften ähneln denen eines dreidimensionalen Gitters. Unabhängig von der Anzahl an Partitionen ergibt sich ein guter relativer Kantenschnitt. Die Benchmarks *pwt* und *brack* sind komplexere geometrische Körper von eher länglicher Struktur. Damit ergibt sich ein geringerer normierter Kantenschnitt, der an einem flachen Anstieg der Messwerte erkennbar ist. Da die geometrisch orientierte Partitionierung mittels raumfüllender Kurven diese strukturellen Eigenschaften nicht berücksichtigt, ergibt sich ein sichtbar größerer normierter Kantenschnitt und damit auch ein hoher relativer Kantenschnitt im Vergleich zu kMetis. Bei wachsender Anzahl an Partitionen sind auch die Schnitte einer optimalen Partitionierung vermehrt gleichmäßig auf das vernetzte Gebiet verteilt, womit die Vorteile einer ausgefeilten Partitionierungsstrategie gegenüber den raumfüllenden Kurven geringer werden.

Der normierte Kantenschnitt der raumfüllenden Kurven hat im Bereich der $|V|^{\frac{1}{2}}$- bis $|V|^{\frac{3}{4}}$-Partitionierung meist einen sichtbaren Hochpunkt. Dagegen steigt der normierte Kantenschnitt von kMetis i. A. nur leicht über den mittleren Knotengrad, dem normierten Kantenschnitt der $|V|$-Partitionierung.

Die Tabellen 3.6 und 3.7 zeigen den Kantenschnitt der 16- und 64-Partitionierungen durch Metis und durch die raumfüllenden Kurven. Das beste Ergebnis von Hilbert- und Lebesgue-Kurve ist jeweils hervorgehoben. Wie auch im zweidimensionalen Fall lässt sich kein großer Unterschied zwischen den Partitionierungsqualitäten der beiden rekursiven raumfüllenden Kurven erkennen. Der relative Unterschied im erzielten Kantenschnitt liegt immer unter 6 Prozent. Dieser ist unerheblich, da beide Varianten schon eine Abweichung von mindestens 50 bis 100 Prozent von der optimalen Lösung haben.

Abbildung 3.25 zeigt den Einfluss der Wahl der Anfangsgeometrie, über welche die raumfüllende Kurve definiert wird (vgl.. Abschnitt 3.2.3). Alle drei Messreihen zeigen den relativen Kantenschnitt der Hilbert-Kurve am Beispiel *pwt*. Die obere Messreihe beginnt mit dem minimalen Quader, der das Netz enthält. Dieser wird rekursiv gemäß der

Netz	*k*Metis	*p*Metis	Hilbert	Lebesgue
pwt	2 992	2 933	**7 605**	7 616
brack	13 225	12 707	29 932	**28 086**
rotor	24 477	23 863	97 609	**93 526**
wave	48 183	48 106	**77 166**	78 592
hermes	119 219	119 170	**256 866**	266 479

Tabelle 3.6: Kantenschnitt der 16-Partitionierungen der dreidimensionalen Benchmarks

Netz	*k*Metis	*p*Metis	Hilbert	Lebesgue
pwt	9 015	9 310	**15 341**	15 674
brack	29 432	29 353	**59 814**	61 054
rotor	52 190	53 623	**176 574**	182 657
wave	94 342	97 010	**130 923**	138 787
hermes	241 771	249 959	**443 450**	473 081

Tabelle 3.7: Kantenschnitt der 64-Partitionierungen der dreidimensionalen Benchmarks

Konstruktionsvorschrift in acht Teile zerlegt. Die zweite Messreihe startet mit einem minimalen Würfel, wobei in diesem Fall der Würfel durch die Erweiterung des minimalen Quaders in positiver Koordinaten-Richtung gegeben ist. Die dritte Messreihe beginnt mit dem minimalen umgebenden Quader und zerlegt ihn bzgl. der in Abschnitt 3.2.3 vorgestellten Strategie zunächst im eindimensionalen, dann im zweidimensionalen und danach im dreidimensionalen Raum (vgl. Abb. 3.6, S. 32).

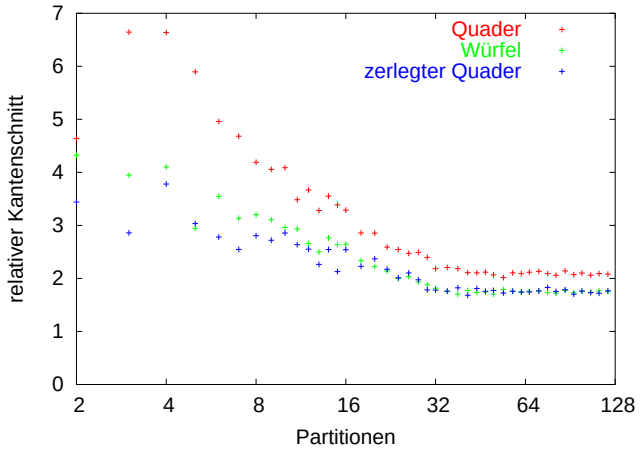


Abbildung 3.25: Vergleich verschiedener Zerlegungsstrategien am Beispiel *pwt*

Es zeigt sich, dass die Art, wie die Zerlegung begonnen wird, einen großen Einfluss auf die Qualität der Partitionierung hat. Wird die Zerlegung mit einem Quader mit deutlich unterschiedlichen Kantenlängen begonnen, bleibt dieses Verhältnis während der rekursiven Zerlegung bestehen. Es entstehen nicht die gewünschten würfelähnlichen Teilkomponenten, so dass die Oberflächen wachsen und damit der Kantenschnitt groß wird. Die relative Größe dieses Nachteils ist von der Granularität der Partitio-

nierung weitgehend unabhängig. Startet die Zerlegung mit einem umgebenden Würfel, entstehen Sprünge in der Kurve, da weite Bereiche des indizierten Gebietes leer sind. Es ergibt sich ein erhöhter Kantenschnitt, dessen absolute Größe von der Anzahl der Partitionen unabhängig ist. Da der gesamte Kantenschnitt mit der Anzahl der Partitionen wächst, fällt gleichzeitig der relative Qualitätsverlust.

Die Untersuchungen an den dreidimensionalen Benchmarks hat gezeigt, dass es Netze gibt, deren Indizierung mittels raumfüllender Kurven zu Partitionierungen mit zu hohem Kantenschnitt führen können. Das Netz *rotor* hat bei der Bisektion, die sich aus der Hilbert-Kurve ergibt, einen relativen Kantenschnitt von 9. Da auch *k*Metis nicht nachweislich die optimale Lösung findet, ist der erzielte Kantenschnitt wahrscheinlich zehn Mal größer als das Optimum. Wie für die anderen Netze auch, fällt der relative Kantenschnitt mit größer werdender Anzahl an Partitionen, er bleibt jedoch im untersuchten Intervall über 3. Dieses auffällig schlechte Verhalten, lässt sich durch die Struktur des Netzes erklären. Es gibt zwei Bereiche im Netz, die eine hohe Verfeinerung aufweisen. Beide haben eine im Wesentlichen eindimensionale Ausdehnung, einer in x-, der andere in y-Orientierung. Teilen die Separatorflächen vom höchsten Level diese Bereiche, ergibt sich an dieser Stelle ein sehr hoher Kantenschnitt.

Bei einer Verschiebung des Separationszentrums im ersten Schritt der Indizierung ergeben sich deutlich bessere Partitionierungen. Dieses Verhalten ist in Abbildung 3.26 anhand der Lebesgue- und der Hilbert-Kurve dargestellt. Für beide Kurven ist die Messreihe mit dem errechneten und mit einem leicht verschobenen Separationszentrum gegeben. Es ergibt sich eine Verbesserung des relativen Kantenschnitts im gesamten untersuchten Bereich. Dies zeigt die prinzipielle Möglichkeit, die Qualität der Indizierung bzgl. der Partitionierung durch vorgeschaltete Heuristiken verbessern zu können.

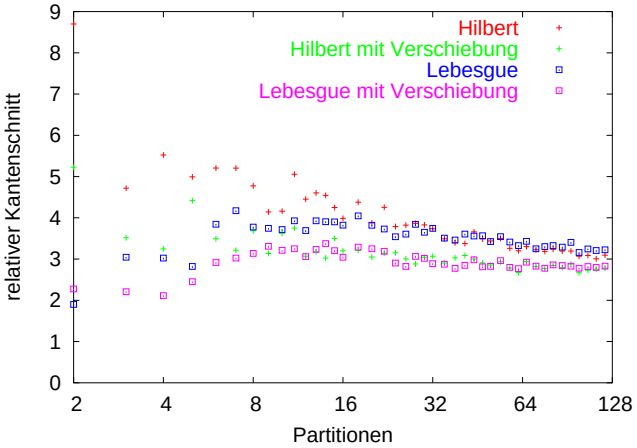


Abbildung 3.26: Einfluss der Wahl des Zentrums auf die Partitionierungsqualität am Beispiel *rotor*

3.4.3.3 Ressourcen

Abbildung 3.27 vergleicht das Laufzeitverhalten der Indizierung durch raumfüllende Kurven mit dem der Partitionierung durch Metis. Die dargestellten Messwerte sind das Mittel aus jeweils zehn Testläufen an dem dreidimensionalen Beispielnetz *hermes*. Die direkte k-Partitionierung mittels Metis (kMetis) zeigt einen leichten Anstieg der Rechenzeit, die rekursive Partitionierung (pMetis) einen deutlichen Anstieg. Bei 128 Partitionen benötigt pMetis über 21 Sekunden und damit etwa viermal so lange wie kMetis (4.3 Sekunden). Dagegen ist die Rechenzeit der vollständigen Knotenindizierung nach der Hilbert- bzw. der Lebesgue-Kurve für alle Partitionierungen mit 1.1 bzw. 0.9 Sekunden konstant.

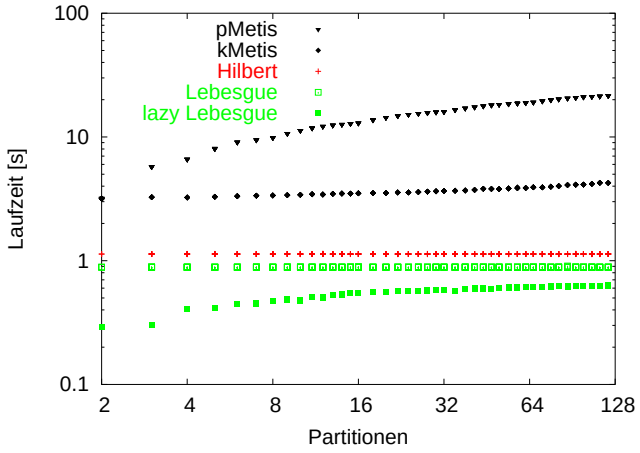


Abbildung 3.27: Laufzeitverhalten der Partitionierungsheuristiken am Beispiel *hermes*

Die Messwerte *lazy Lebesgue* geben die Laufzeit einer unvollständigen Indizierung bzgl. der Lebesgue-Kurve an. Dabei wird ein quadratisches Teilgebiet nur dann weiter zerlegt, wenn die enthaltenen Knoten verschiedenen Partitionen zuzuordnen sind. Liegt in dem Teilgebiet ein Intervall an Knoten, das vollständig Bestandteil einer einzelnen Partition ist, wird die Rekursion abgebrochen. Die innere Ordnung dieser Knoten ist für eine statische Partitionierung unwichtig und es erspart für kleinere Anzahlen an Partitionen bis zu 70 Prozent Rechenzeit.

Die Untersuchungen zeigen, dass die raumfüllenden Kurven deutlich schneller zu einer Partitionierung führen als Metis. Ist für eine FE-Applikation eine statische Partitionierung ausreichend, ist die Partitionierungszeit in der gegebenen Größenordnung gegenüber der Gesamtrechenzeit vernachlässigbar. Dies ändert sich, wenn die charakteristischen Eigenschaften einer Applikation eine regelmäßige Repartitionierung erfordern, wie in Abschnitt 5.2 beschrieben. In diesem Falle haben die Zeiten der Repartitionierung möglicherweise einen großen Einfluss auf die Gesamtrechenzeit, sie können diese mitunter dominieren.

Neben der Laufzeit ergibt sich auch ein Vorteil bei dem Speicherbedarf der untersuchten Partitionierungsmethoden. Während Metis für die Partitionierung des Netzes *hermes* etwa 200 MByte Speicher benötigt liegt der Bedarf der raumfüllenden Kurven mit rund

5 MByte deutlich niedriger.

3.4.4 Zusammenfassung

Die Qualität einer Partitionierung über raumfüllende Kurven ist stark von der geometrischen Struktur des Netzes abhängig. Dabei sind sowohl die Form des diskretisierten Gebietes als auch die Form von Löchern oder verfeinerten Bereichen von Bedeutung. Haben diese eine Ausdehnung in niedrigerer Dimension als das Gesamtnetz, ergeben sich häufig optimale Partitionierungen mit kleinem normierten Kantenschnitt bei geringer Partitionsanzahl. Das Partitionierungswerkzeug Metis findet eine gute Lösung, da es die Graphstruktur berücksichtigt. Dagegen haben die k-Partitionierungen der raumfüllenden Kurven einen normierten Kantenschnitt auf einem gleich bleibenden Niveau. Bei einer größeren Anzahl an Partitionen verringert sich der Einfluss der geometrischen Eigenschaften zunehmend und die Qualität der Lösungen von Metis und raumfüllenden Kurven erreicht für viele Netze eine vergleichbaren Wert. Im Rahmen der untersuchten Benchmark-Netze liegt der relative Kantenschnitt für zweidimensionale Netze zwischen 1.3 und 1.5 und für dreidimensionale zwischen 1.3 und 2. Der geringste Qualitätsverlust gegenüber Metis wird im Gitter erreicht, dessen Struktur der Konstruktion der rekursiven raumfüllenden Kurven entspricht.

Der Vorteil einer ausgefeilten Partitionierungsheuristik gegenüber den geometrischen Verfahren, die auf raumfüllenden Kurven basieren, ist insbesondere bei kleiner Partitionsanzahl groß. Um eine Indizierung zu finden, die für eine beliebige Anzahl an Partitionen befriedigende relative Kantenschnitte liefert, könnte eine Kombination beider Varianten sinnvoll sein. Zunächst wird für ein geeignetes k eine Partitionierung mit sehr geringem Kantenschnitt gesucht und innerhalb der Partitionen eine Indizierung über raumfüllende Kurven durchgeführt.

Eine zweite Variante zur Verbesserung könnte die geeignete Wahl der Lage der Separatoren vom höchsten Level darstellen. Am Beispiel *rotor* ist der Einfluss der Lage des Separatorzentrums auf den normierten Kantenschnitt gezeigt worden. Ebenso könnte eine geschickt gewählte Rotation der raumfüllenden Kurve um das Separatorzentrum einen ähnlich positiven Einfluss auf den Kantenschnitt haben, wie zwischen den einfachen geometrischen Verfahren RCB und RIB (vgl. Abschnitt 3.1). Letzteres kann in vielen Netzen eine bessere Partitionierung erzielen, da sie die freie Lage der Separatorlinie im Raum erlaubt.

Kapitel 4

Effiziente globale Kontaktsuche

Bei der Simulation strukturmechanischer Problemstellungen kommt es zu gegenseitigen Durchdringungen der sich bewegenden Netze. Aus Randbedingungen und aus inneren Kräften an Knoten des Netzes, die aus dem Materialverhalten resultieren, ergeben sich Verschiebungen der Knoten. Diese führen zu Überschneidungen von Netzen unterschiedlicher Objekte oder Überschneidungen innerhalb eines Netzes. Diese Durchdringungen stellen nichtphysikalische Situationen dar, die innerhalb der Simulation korrigiert werden müssen. In Abbildung 4.1 ist die Durchdringung und die danach folgende Korrektur an einem Beispiel dargestellt. Die Verschiebungsvektoren der Knoten im aktuellen Zeitschritt sind blau, die resultierenden Korrekturkräfte rot markiert.

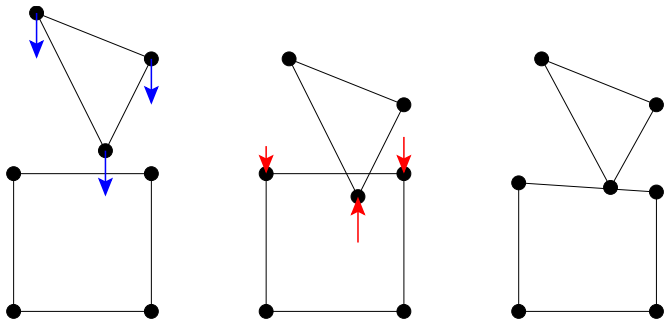


Abbildung 4.1: Nichtphysikalische Durchdringung und Kontaktkorrektur

Dieses Kapitel widmet sich der Phase der höchsten algorithmischen Komplexität, der Bestimmung der Kontaktpaarungen. Zunächst wird die Einordnung dieser Phase in den allgemeinen Ablauf der Kontaktbehandlung dargestellt und verschiedene Methoden der Kontaktkorrektur skizziert. In Abschnitt 4.3 und 4.4 wird eine verbesserte zellbasierte Methode zur globalen Kontaktsuche auf Basis der Lebesgue-Kurve vorgestellt und die Kosten anhand eines Szenarios analysiert. In Abschnitt 4.5 wird ein Maß über raumfüllende Kurven definiert, welches dem Vergleich verschiedener raumfüllender Kurven innerhalb der globalen Kontaktsuche dient.

4.1 Kontaktbehandlung in der expliziten FE-Simulation

Unabhängig von der Art der FE-Simulation und dem Verfahren der Kontaktkorrektur gliedert sich die Kontaktbehandlung i. A. in die Phasen *globale Kontaktsuche*, *lokale Kontaktsuche* und *Kontaktkorrektur*. Die letzte Phase ist von der Art der Simulation und dem gewählten Korrekturverfahren abhängig. Bei der impliziten FE-Simulation erfolgt die Korrektur durch eine Erweiterung der Steifigkeitsmatrix. Für jeden Kontaktpunkt wird ein Eintrag erstellt, der die Wechselwirkung der betroffenen Oberflächensegmente beschreibt. Damit sind die Kräfte aus dem Inneren des simulierten Materials und die Kräfte aus den Kontaktsituationen in einem Gleichungsystem beschrieben und werden gemeinsam bestimmt. Bei der expliziten FE-Simulation werden die Phase der Kraftberechnung im Inneren des Materials aufgrund der Spannungs- und Dehnungszustände und die Phase der Kontaktkorrektur nacheinander ausgeführt. Innerhalb der Schleife zur Zeitintegration werden jeweils beide Phasen durchlaufen. Aufgrund der i. A. sehr geringen Zeitschrittweite ist der Fehler der nachlaufenden Kontaktbehandlung als gering anzunehmen.

Für beide Arten der FE-Simulation müssen vor der Kontaktkorrektur zunächst die erforderlichen Parameter bestimmt werden. Diese hängen von dem Verfahren der Kontaktkorrektur ab. In jedem Fall benötigt diese Phase Informationen über die Kontaktpaarungen, die Penetrationstiefe und die Korrekturrichtung. Diese werden in der Phase der lokalen Kontaktsuche (engl.: *local search* oder *post-contact search*) bestimmt. Je nach Anwendungsgebiet und Bedeutung der Kontaktbehandlung für die gesamte Simulation ist ein mehr oder weniger komplexer Entscheidungsmechanismus notwendig, der nach der Feststellung eines Kontaktes von Oberflächensegmenten des FE-Netzes die Korrekturrichtung festlegt. Aus dieser ergibt sich die zugehörige Kontaktpaarung und die Penetrationstiefe.

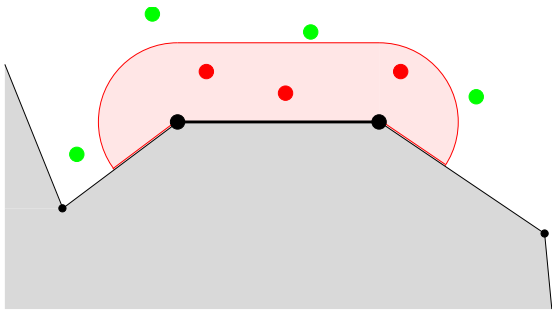


Abbildung 4.2: Kontaktkandidaten (rot) zu einem Oberflächensegment

Die höchste algorithmische Komplexität ergibt sich in der globalen Kontaktsuche (engl.: *global search* oder *pre-contact search*). Diese Phase bestimmt die Menge der potentiellen Kontaktpaarungen innerhalb der nächsten Zeitschritte. Zu jedem Oberflächensegment werden die Segmente bestimmt die in einer begrenzten Distanz liegen (vgl. Abb. 4.2). Wie häufig diese Phase durchzuführen ist, ist von der gewählten Distanz und der Bewegungsgeschwindigkeit des Netzes abhängig. Die regelmäßige Durchführung dieser Pha-

se ist notwendig, wenn die Menge der potentiellen Kontaktpaarungen zu groß ist, um sie vor der Berechnung vollständig bestimmen und in der lokalen Kontaktsuche regelmäßig auf Kontakt testen zu können. Es ist offensichtlich, dass ein paarweiser Test aller Oberflächensegmente bzgl. ihrer geometrischen Lage zueinander einen zu großen Rechenaufwand bedeutet.

4.2 Methoden der Kontaktsuche

Alle Methoden zur globalen Kontaktsuche basieren auf der Idee, die gegebenen Objekte in einer Datenstruktur so zu ordnen, dass innerhalb dieser effiziente Kontaktanfragen möglich sind. Dazu wird der gesamte Suchraum in kleine Gebiete zerlegt. Dabei gibt es zwei unterschiedliche Ansätze zur Strukturierung des Suchraumes, über die sich die Verfahren klassifizieren lassen. Entweder findet eine geometrische Ordnung der einzelnen Punkte (Knoten des Netzes) oder eine Partitionierung der gegebenen Objekte statt.

4.2.1 Geometrische Ordnung

Die meisten Datenstrukturen zur geometrischen Ordnung sind von dem verwandten Problem des *orthogonal range searching* abgeleitet. Dabei besteht die Aufgabe darin, aus einer gegebenen Punktmenge alle die Punkte zu finden, die in einer beliebigen achsenparallelen Box liegen. Auch wenn zu dieser Fragestellung viele Ergebnisse existieren, die im Wesentlichen das Verhalten im schlechtesten Fall analysieren [1, 9, 51], ist wenig über deren Effizienz in praktischen Anwendungen bekannt. Die Datenstrukturen und Algorithmen, die bewiesenen oberen Schranken zu Grunde liegen, sind häufig zu komplex und die Konstanten zu den asymptotischen Kosten zu groß für einen praktischen Einsatz in Applikationen, insbesondere in Räumen mit Dimension $d > 2$ [1, 16, 26].

Die meisten der eingesetzten Datenstrukturen basieren auf Varianten von *kd-Bäumen*. Weitere wichtige Methoden sind *Projektion*, *Zellen* und *Intervall-Bäume* (range-trees), die alle im Folgenden kurz beschrieben werden [17, 26].

kd-Bäume Der gesamte Suchraum wird über die Struktur eines Baumes aufgeteilt. Dabei entspricht jeder innere Knoten des Baumes einer Schnittebene, welche orthogonal zu einer der Achsen verläuft. Jeder Knoten teilt den verbliebenen Suchraum in zwei Teile. Die Aufteilung wird soweit durchgeführt, bis ein Knoten des Baumes nur eine konstante Anzahl Punkte im Raum repräsentiert.

Für eine Anfrage wird in jeder Ebene des Baumes getestet, auf welcher Seite der Schnittebene die Region liegt. Liegt die Region vollständig auf einer Seite der Schnittebene fährt die Anfrage am entsprechenden Knoten fort. Wird die Region von der Ebene geschnitten, müssen beide Söhne des Knotens abgearbeitet werden. An einem Blatt des Baumes wird jeder enthaltene Punkt gegen die Anfrageregion getestet. Abbildung 4.3 zeigt die Aufteilung einer Punktmenge durch einen kd-Baum. Zu einer Anfrage ist ein

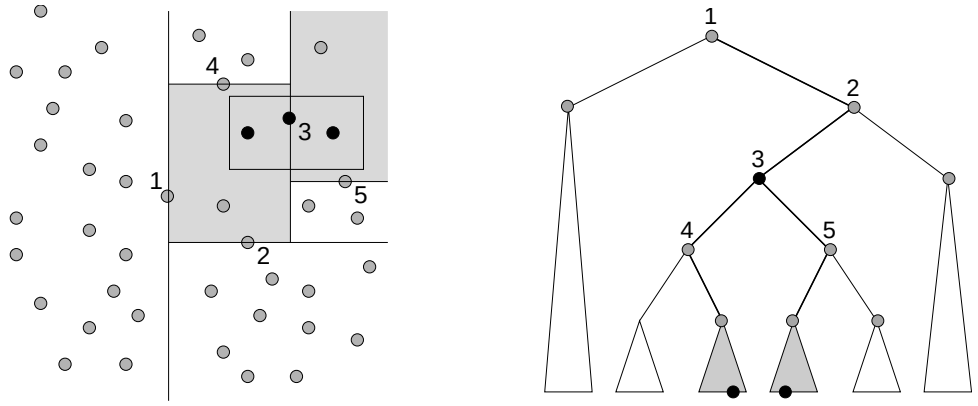


Abbildung 4.3: Strukturierung des Raumes durch einen kd-Baum

Teil der Pfade angegeben, über die der Baum durchlaufen wird. Die von den grauen Teilbäumen repräsentierte Fläche ist hervorgehoben.

Es ist offensichtlich, dass die Effizienz dieser Struktur von der Wahl der Schnittebenen abhängig ist. Oftmals wird die Ebene orthogonal zur größten Ausdehnung der verbliebenen Punktmenge gewählt, an der Position, die diese Menge halbiert. Ursprünglich war die Bezeichnung kd-Baum für eine Baumstruktur zur Aufteilung des k-dimensionalen Raumes eingeführt worden. Mittlerweile wird der Name jedoch für die allgemeine Idee verwendet, so dass der 2-d-Baum als zweidimensionaler kd-Baum bezeichnet wird.

Projektion Zur Initialisierung dieser Methode wird für jede Dimension eine Liste der Punktmenge erstellt. Jede Liste wird bzgl. der Koordinaten ihrer Dimension geordnet. Für die Beantwortung einer Anfrage wird in jeder Liste das Intervall gesucht, das die Anfrageregion überdeckt. Das kürzeste so gefundene Intervall wird genutzt, um die darin enthaltenen Punkte gegen die verbliebenen $d-1$ Dimensionen der Anfrageregion zu testen. Die Suche der Intervalle in jeder Dimension ist in Zeit $O(\log n)$ möglich. Bei einer gleichverteilten Punktmenge kann die Komplexität der Anfrage um den Faktor $\sqrt[d]{n}$ verringert werden. Die Analyse im schlechtesten Fall ergibt jedoch Kosten in Höhe von $O(n)$ für eine rechtwinklige Anfrageregion.

Zellen Für diese Methode wird der gesamte Suchraum in m gleichgroße Zellen aufgeteilt. Die Aufteilung wird üblicherweise relative zur Größe n der Punktmenge gewählt, d. h. $m = n/c$, wobei c eine geeignete positive Konstante ist. Für jede dieser Zellen wird eine Liste aller enthaltenen Knoten aufgestellt. Dabei wird ein direkter Zugriff auf jede einzelne Zelle sichergestellt, z. B. über ein mehrdimensionales Feld. Für eine Anfrage wird festgestellt, welche der Zellen geschnitten werden. Für alle Punkte, die diesen Zellen zugeordnet sind, wird überprüft, ob sie in der gegebenen Region enthalten sind.

Diese Methode ist insbesondere bei einer gleichmäßigen Verteilung der Punkte geeignet. Bei einer ungleichmäßigen Verteilung ist es notwendig, die Anzahl der Zellen soweit zu erhöhen, dass nur eine kleine Anzahl an Punkten in einer Zelle liegt. Wenn es im

vorhinein bekannt ist, kann auch die kleinste Größe einer Anfrageregion ein Maßstab für die Ausdehnung der Zellen sein. Wird dabei m deutlich größer als n ist es nicht mehr möglich, eine Struktur mit direktem Zugriff zu wählen. Ein effizienter Zugriff kann dann z. B. durch eine Hash-Tabelle erreicht werden.

Intervall-Bäume Intervall-Bäume kombinieren die Idee des *divide-and-conquer* Ansatzes beim kd-Baum mit der Strategie der Sortierung bzgl. der Koordinaten aller Achsen bei der Projektion. Die Anfrage im eindimensionalen kd-Baum resultiert in zwei Pfaden von der Wurzel zu den Blättern. Die bis zu $2\log_2 n$ Teilbäume zwischen den beiden Pfaden enthalten die Punkte, die in der Anfrageregion liegen. Der eindimensionale Intervall-Baum entspricht dem eindimensionalen kd-Baum. Er ist gleichzeitig die Grundlage des mehrdimensionalen Intervall-Baumes. Zu jedem Knoten dieses Hauptbaumes (*first level tree*) wird ein eindimensionaler kd-Baum aufgebaut, der die Punkte bzgl. der zweiten Koordinate ordnet, die im Teilbaum zu diesem Knoten liegen. Diese assoziierten Nebenbäume (*second level tree*) werden über Zeiger mit dem Hauptbaum verbunden. Für jede weitere Dimension werden zusätzliche Nebenbäume erstellt, die zu Knoten der Nebenbäume des nächsthöheren Levels assoziiert sind. Damit erhöht sich die Größe der Datenstruktur für jede Dimension um den Faktor $\log_2 n$.[1]

Eine Anfrage erfolgt zunächst über den Hauptbaum. Von der Wurzel beginnend werden alle Knoten durchlaufen, deren Teilbäume die Anfrageregion schneiden. Wird ein Knoten gefunden, dessen Teilbaum bzgl. der ersten Dimension vollständig in der Anfrageregion liegt, wechselt der Suchpfad in den assoziierten Nebenbaum. Dort wird die Anfrage bzgl. der zweiten Dimension weitergeführt. Alle so gefundenen Blattknoten in Nebenbäumen vom niedrigsten Level liegen in der Anfrageregion.

In Abbildung 4.4 ist die Ordnung einer Punktmenge im Intervall-Baum skizziert. Der Hauptbaum ist ein Binärbaum mit einer Ordnung bzgl. der x-Koordinaten. Der dargestellte Nebenbaum ordnet die Punkte gemäß ihrer y-Koordinaten, die zwischen den Separationslinien durch Punkt 1 und 2 liegen. Die Anfrageregion enthält dieses Intervall an x-Koordinaten vollständig, so dass die Anfrage in diesen Nebenbaum verzweigt, während der zweite Pfad der Anfrage im Hauptbaum weiterläuft. Die Regionen, die von den beiden grauen Teilbäumen repräsentiert werden, sind in der linken Darstellung hervorgehoben.

Die Kosten für die Initialisierung ($I(n)$) und Anfrage ($Q(n)$) und die Größe der Datenstruktur ($S(n)$) für die vier vorgestellten Basismethoden sind in Tabelle 4.1 angegeben [17, 26]. Dabei ist n die Größe der Punktmenge im d-dimensionalen Raum und m die Anzahl der Punkte in der Anfrageregion. Zwischen den drei Methoden Zellen, kd-Baum und Intervall-Baum lässt sich ein Tradeoff zwischen der Zeit für die Initialisierung und die der Anfrage feststellen. Zudem ist der Speicherbedarf des Intervall-Baumes größer als bei den anderen Verfahren.

[1] Jeder Knoten ist indirekter Sohn von $\log_2 n$ Knoten des Hauptbaumes. Damit taucht er auch in $\log_2 n$ Nebenbäumen der zweiten Dimension auf, so dass sich die Gesamtgröße um den Faktor $\log_2 n$ erhöht.

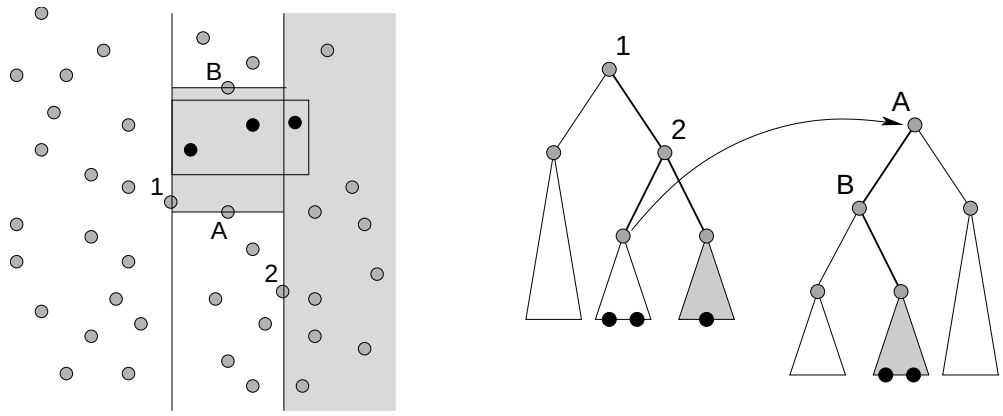


Abbildung 4.4: Strukturierung des Raumes durch einen Intervall-Baum

Methode	$I(n)$	$S(n)$	$Q(n)$
Projektion	$O(n \log n)$	$O(n)$	$O(n+m)$
Zellen	$O(n)$	$O(n)$	$O(n+m)$
kd-Baum	$O(n \log n)$	$O(n)$	$O(n^{(\frac{d-1}{d})}+m)$
Intervall-Baum	$O(n \log^{d-1} n)$	$O(n \log^{d-1} n)$	$O(\log^d n + m)$

Tabelle 4.1: Kosten im schlechtesten Fall im d-dimensionalen Raum ($d \in \{2,3\}$)

Eine Analyse der vier Methoden im mittleren Fall in einem festgelegten Szenario liefert ein deutlich anderes Ergebnis [26]. Seien die Punkte im Suchraum gleichmäßig verteilt. Unter der Annahme, dass eine Anfrage der Größe s^3 eine konstante Anzahl r an Punkten enthält ergeben sich die in Tabelle 4.2 angegebenen Kosten. Die Reduktion der Kosten für die Projektion und die Zellen ergibt sich aus der Annahme einer gleichverteilten Punktmenge. Damit ist die Reduktion der Kandidaten um den Faktor $\sqrt[d]{n}$ nach dem Test gegen eine der Dimensionen garantiert. Bei der Zellen-Methode ist die Anzahl der durchsuchten Zellen und die Anzahl der darin enthaltenen Punkte konstant.

Methode	$\bar{Q}(n)$
Projektion	$O(n^{(\frac{d-1}{d})})$
Zellen	$O(1)$
kd-Baum	$O(\log n)$
Intervall-Baum	$O(\log^d n)$

Tabelle 4.2: Kosten im mittleren Fall im d-dimensionalen Raum ($d \in \{2,3\}$)

Die Reduktion der Kosten für den kd-Baum ergibt sich aus der konstanten Größe der Anfrageregion, die aus den anderen Randbedingungen folgt. Für den Beweis der oberen Schranke ist es notwendig, alle durchlaufenen Knoten zu zählen, deren zugehöriger

Bereich des Suchraumes von der Anfrageregion geschnitten wird, aber nicht vollständig in ihr enthalten ist. Im schlechtesten Fall werden nicht die Schnitte der Kanten eines endlichen Rechtecks analysiert, sondern die Schnitte von Linien. Insgesamt resultieren die deutlich niedrigeren Kosten eher aus dem starren Szenario, als aus dem Schritt vom schlechtesten Fall zum mittleren Fall.

Diese Ergebnisse werden in einer vergleichenden Evaluation der vier Methoden an praktischen Eingabeinstanzen grundsätzlich bestätigt [26]. Dort wird gezeigt, dass zur Lösung des *range searching* die Strukturierung der Objekte über Zellen und kd-Bäume den anderen beiden Varianten überlegen ist. Die Vorteile der Zellen-Methode liegen in der geringen Zeit für den Aufbau der Datenstruktur und die geringe Anzahl an Tests gegen die Anfrageregion aufgrund der feinen Ordnung der Punkte. Der kd-Baum hat geringfügig höhere Laufzeiten bei der Initialisierung, dafür ist die Effizienz jedoch unabhängig von der Verteilung der Punkte im Raum.

Für die Methoden des kd-Baumes und der Zellbildung gibt es eine Reihe von Erweiterungen und Verbesserungen, die sie im praktischen Einsatz effektiver machen. Für die Strukturierung im kd-Baum sind verschiedene Optimierungen hinsichtlich der Speicherung der Daten und der Ordnung der Objekte im Baum vorgestellt worden (z. B. *alternating digital tree* [11] und *augmented spatial digital tree* [23]).

Bei Methoden, die im Bereich der konkreten Anwendung entstanden sind, finden sich Kombinationen oder Abstraktionen aus den Basismethoden des range searching. Die Kombination von Zellen und Bäumen ist z. B. im Octree (bzw. Quadtree) gegeben, in dem der Suchraum hierarchisch in regelmäßige Bereiche zerlegt wird [32, 34, 70]. An den Blättern liegen die Inhalte der Zellen, während die Anfrage der in einem kd-Baum gleicht. In diesem Fall ist die Effizienz des Verfahrens wiederum von der gleichmäßigen Verteilung der Punkte im Raum abhängig. Diese Methode findet sich vielfach in Anwendungen zur Simulation von Festkörpern, wie Robotern oder im CAD-Umfeld.

Im Bereich der FEM ist eine weitere Methode vorgestellt worden, die zur Klasse der zellbasierten Verfahren gehört. Der Position-Code-Algorithmus kann als Kombination aus Zellen- und Projektionsmethode angesehen werden [59]. Auf die sehr große Anzahl an Zellen wird nicht über eine Hashing-Tabelle zugegriffen, sondern sie sind bzgl. der Koordinaten sortiert, so dass sich eine Verwandtschaft zur Projektion ergibt. Die Ordnung verhält sich aber auch ähnlich zu einem degenerierten kd-Baum: Die Knoten der oberen Ebenen repräsentieren nur Schnittebenen orthogonal zu einer der Dimensionen. In den folgenden Ebenen finden sich Knoten zu Schnittebenen zur zweiten Dimension usw. Das Verfahren wird in Abschnitt 4.3 näher beschrieben und dient als Grundlage für die hier vorgestellte Methode einer zellbasierten Datenstruktur.

4.2.2 Objekt-Partitionierung

Für einen vorläufigen Kontakttest werden um komplexe Körper einfacherer geometrische Strukturen gelegt. Die Körper können nur dann zueinander in Kontakt getreten sein, wenn sich die umschließenden Objekte schneiden. Der Test dieser geometrisch

weniger komplexen Objekte ist einfacher und damit schneller ausführbar. Aus dieser Beobachtung folgt die Idee der Objekt-Partitionierung, in der ursprünglich eine Aufteilung der Objekte über vier Ebenen vorgeschlagen wurde [85]: In der ersten Ebene existiert eine Aufteilung auf die Körper, in der zweiten eine Gruppierung in die einzelnen Oberflächen eines Körpers, in der dritten und vierten eine Aufteilung in Segmente bzw. Knoten. In jeder Ebene findet ein vorläufiger Kontakttest statt, indem umgebende achsenparallele Quader (bzw. Rechtecke) auf einen Schnitt getestet werden. Sind die Regionen disjunkt, ist kein weiterer Test erforderlich. Andernfalls wird der Test mit allen Teilobjekten der folgenden Ebene paarweise durchgeführt. Für weit entfernt liegende Objekte endet die Suche schnell, so dass nur tatsächliche Kontaktsituationen zu einem höheren Aufwand führen. Eine Verallgemeinerung der Hierarchie auf Binärbäume führt zu den BV-Bäumen (*bounding volumes*) [8]. BV-Bäume, deren umschließende Geometrien lediglich aus Ebenen orthogonal zu den Achsen verlaufen, werden als AABB-Bäume (*axis aligned bounding boxes*) bezeichnet.

Die Wahl der geometrischen Objekte, die für den vorläufigen Kontakttest eingesetzt werden, haben einen großen Einfluss auf die Laufzeit des Verfahrens. Wichtige Kriterien sind (a) ein schneller Test auf Schnitt, (b) gute Approximation der (Teil-)Objekte, um viele Kombinationen ohne Kontakt schon auf oberen Ebenen des Baumes herausfiltern zu können und (c) eine schnelle Neuberechnung der Geometrie, wenn sich das enthaltene Objekt bewegt hat.

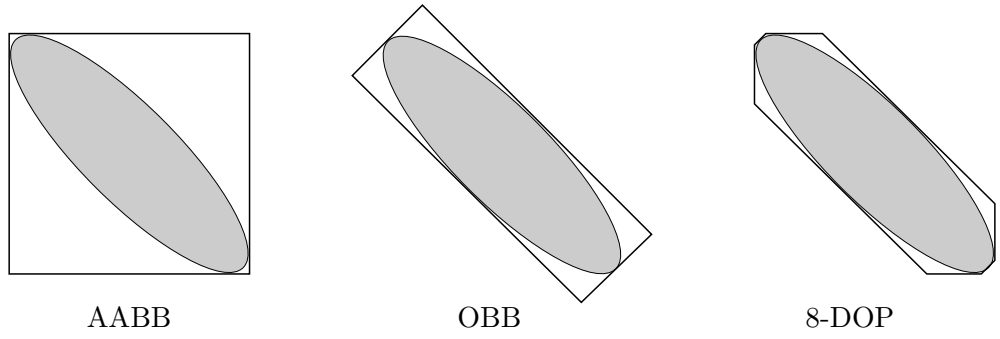


Abbildung 4.5: Exemplarischer Vergleich unterschiedlicher umschließender Volumen

Abbildung 4.5 stellt die AABB und zwei weitere Arten von umschließenden Volumen im zweidimensionalen Raum dar. In OBB-Bäumen (oriented bounding boxes) werden auch rechtwinklige Objekte für den Kontakttest eingesetzt, ihre Orientierung ist jedoch nicht von den Achsen, sondern von der Form des Objektes abhängig. Damit ist eine deutlich geringere Abstraktion von der Form der Objekte möglich, und das umschlossene Volumen verringert sich. Eine vergleichende Analyse von AABB und OBB zeigt, dass letztere Variante in vielen Bereichen der Computer-Graphik überlegen ist. Die bessere Näherung der gegebenen Geometrien reduziert die Anzahl der notwendigen Schnitttests stärker als die Kosten für einen einzelnen Test steigen [29].

Eine weitere Verallgemeinerung stellt die Näherung der Objekte durch Polytope aus k Ebenen dar, die als k-DOP (*discrete oriented polytope*) bezeichnet werden. Im Gegensatz zu den OBB's sind diese k Ebenen nicht von der Ausrichtung des einzelnen Ob-

jektes abhängig, sondern für alle Objekte identisch. Untersuchungen haben gezeigt, dass die bessere Annäherung über komplexere Objekte nicht zu der gewünschten deutlichen Reduktion der Anzahl von Tests zum Schnitt zweier Objekte führt [48]. BV-Bäume sind in verschiedenen Softwarepaketen zur Kontakterkennung implementiert. Zu den bekanntesten gehören H-COLLIDE (OBB-basiert) [30], SOLID (AABB-basiert) [77] und SiLVIA (BV wählbar) [49].

Die Objekt-Partitionierung wird hauptsächlich im Bereich der Virtual Reality (VR) eingesetzt. Die zumeist wenigen Objekte lassen sich oftmals durch wenige geometrische Grundformen zusammensetzen (wie Straßen, Häuser oder Möbel). Zudem ergeben sich in diesen Umgebungen relativ wenige Kontaktsituationen, so dass vielfach ein erfolgreicher Ausschluss von potentiellen Kontaktpaarungen bereits in den oberen Ebenen des Baumes möglich ist. Bei FE-Simulationen im Bereich der Strukturmechanik ergeben sich häufig viele Gebiete, in denen Oberflächen sehr nahe beieinander liegen. Damit führen viele Kontaktanfragen zu den Blättern des BV-Baumes, so dass im schlechtesten Fall ein quadratischer Aufwand bzgl. der Netzgröße erreicht wird. Zudem müssen für eine Simulation physikalischer Vorgänge alle Kontaktsituationen erkannt werden, während in anderen Bereichen wie der VR teilweise nur die Existenz eines Kontaktes von Bedeutung ist. In [50] und [20] wird gezeigt, dass die hierarchische Partitionierung und Ordnung der Objekte für einige Anwendungen der FE-Simulation nicht geeignet ist. In diesen treten häufig Situationen auf, die für diese Methode den schlechtesten Fall ergeben (vgl. [74, 86]).

4.2.3 Parallele Kontaktsuche

In Simulationen zur Strukturmechanik beansprucht die Kontaktbehandlung häufig einen großen Anteil der gesamten Rechenzeit. Damit ist für eine gute Skalierung einer FE-Applikation eine Parallelisierung der Kontaktbehandlung erforderlich. Die Berechnungskosten der Material- und Kontaktmodellierung sind nicht gleich im Netz verteilt. So ist die Kontaktbehandlung nur für Oberflächen erforderlich, während die Materialberechnung in allen Elementen stattfindet. Zudem hat die Verteilung der Oberflächen je nach eingesetztem Verfahren zur Kontaktsuche einen Einfluss auf deren Kosten, wie die obige Diskussion gezeigt hat. Damit ist für die Kontaktbehandlung im Allgemeinen eine Partitionierung erforderlich, die sich gänzlich von einer guten Graphpartitionierung unterscheidet, die mit den in Kapitel 3 vorgestellten Methoden bestimmt werden kann.

In [65, 12] wird für die FE-Berechnung eine statische Aufteilung mittels Chaco [37], einem allgemeinen Werkzeug zur Graphpartitionierung, vorgeschlagen. Davon unabhängig werden die Oberflächen über die *Recursive Coordinate Bisection* (RCB) aufgeteilt (vgl. Abschnitt 3.1). Für das geometrische Problem der Kontaktsuche ist die Aufteilung über ein geometrisches Verfahren naheliegend. Die Anzahl der erforderlichen Informationen von anderen Recheneinheiten während der Kontaktsuche ist gering, so dass für mehrere Benchmarkinstanzen ein gutes Skalierungsverhalten erzielt wird.

Durch die Entkopplung in zwei Partitionierungsaufgaben entsteht eine neue Problematik. Die Informationen zu den Oberflächensegmenten müssen vielfach auf zwei unterschiedlichen Rechenknoten gespeichert sein: auf dem, welcher für die FE-Berechnung des angrenzenden Elementes verantwortlich ist und auf dem, welcher die Kontaktbehandlung für das Segment ausführt. Damit ist zwischen den Phasen der Berechnung innerer Kräfte und der Kontaktbehandlung ein Austausch von Informationen erforderlich. Dieser führt im schlechtesten Fall zu einer All-to-all-Kommunikation.

Dieser zusätzliche Kommunikationsaufwand wird vermieden, wenn eine Partitionierung bestimmt wird, welche die Rechenlast für beide Phasen der FE-Simulation gleichmäßig aufteilt (*multi-constraint graph partitioning*) [44]. Dazu ist eine komplexe Beschreibung des Optimierungsproblems erforderlich. Zum einen müssen die Gewichte der beiden Phasen berücksichtigt werden, die es gleichmäßig zu verteilen gilt. Zum anderen soll der Kommunikationsaufwand minimiert werden, der sich aus dem Kantenschnitt des Graphen (Kommunikation während der FE-Phase) und dem Schnitt der virtuellen Kanten zwischen den Oberflächenknoten (Kommunikation während der Kontaktphase) ergibt.

Um die globale Kontaktsuche über die Partitionsgrenzen hinweg effizient ausführen zu können, wird ein Suchbaum (*decision tree*) aufgebaut. In diesem ist der Raum in rechtwinklige Bereiche geteilt, die jeweils nur Knoten einer Partition enthalten. Die Struktur des Baumes ist mit der eines kd-Baumes vergleichbar, wobei in diesem Fall die Tiefe stark von dem Verlauf der Partitionsränder abhängig ist. Verlaufen die Partitionsgrenzen diagonal, ist eine sehr feine Aufteilung in rechtwinklige Bereiche erforderlich, die zu einem tiefen Suchbaum führen. Um die Tiefe des Suchbaumes zu verringern, wird auf Basis der gefundenen Partitionierung zunächst ein vorläufiger Suchbaum aufgebaut, dessen Blätter auch solche Regionen enthalten können, die Knoten unterschiedlicher Partitionen enthalten. Mittels lokaler Austauschheuristiken werden die Knoten des Graphen solange ausgetauscht, bis der Suchbaum den geforderten Kriterien entspricht.

4.3 Algorithmus zur globalen Kontaktsuche

In diesem Abschnitt wird für die Aufgabe der globalen Kontaktsuche eine verbesserte Version des Position-Code-Algorithmus vorgestellt [19]. Es handelt sich um eine zellbasierte Methode mit sehr feiner Aufteilung und zählt zu der Klasse der Verfahren mit geometrischer Ordnung. Für den Algorithmus wird angenommen, dass die größte Verschiebung eines Oberflächenknotens im Netz beschränkt und bekannt ist. In Abhängigkeit von dieser Verschiebung ($\Delta_s^{\max}$) und der Schrittweite Δt wird ein Bereich um jedes Oberflächensegment definiert, der *Halo*. Alle Knoten die in dem Halo eines Segmentes liegen, sind Kontaktkandidaten bzgl. dieses Segmentes für die kommenden Zeitschritte (vgl. Abb. 4.2, S. 66).

4.3.1 Linearer Position-Code

Für die Bestimmung der Knoten innerhalb eines Halos wird eine künstliche Ordnung über alle Oberflächenknoten eingeführt. Dazu wird das Berechnungsgebiet in kleine Zellen unterteilt, die jeweils eine eindeutige Nummer erhalten. Jedem Knoten wird die Nummer der Zelle zugewiesen, in der er liegt. Ist die Menge der Knoten bzgl. dieser Nummerierung sortiert, können zu zwei gegebenen Knoten eines Segmentes effizient alle Knoten in seinem Halo bestimmt werden.

Dieses Prinzip wurde von Oldenburg und Nilsson eingeführt [59].[2] Sie haben eine zeilenweise Ordnung vorgeschlagen, mit der höchsten Priorität auf der z-Koordinate und der niedrigsten Priorität auf der x-Koordinate. In Abbildung 4.6 ist die Nummerierung der Zellen und die Anfrage der Kontaktkandidaten zu einem Oberflächensegment im zweidimensionalen Raum skizziert. Die Separationslinien der Zellen sind schwarz. Die Reihenfolge der Knoten des FE-Netzes ist durch die blaue Linie skizziert. Alle Knoten die gegen den Halo getestet werden müssen, liegen in dem rot markierten Bereich, der das achsenparallele Rechteck um das Segment, den *Kontaktbereich*, überdeckt. Diese Überdeckung wird mit vier Intervallen der gesamten Knotenmenge erreicht. Die Reduktion der mehrdimensionalen Knotenkoordinaten auf eine eindimensionale Ordnung erlaubt den Einsatz effizienter Sortier- und Suchalgorithmen.

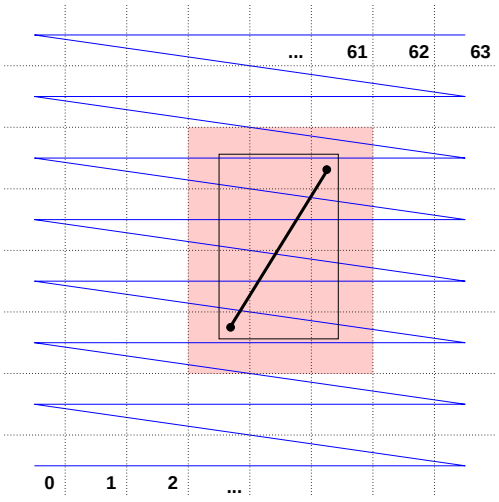


Abbildung 4.6: Suche der Kontaktkandidaten im Position-Code-Algorithmus

Die regelmäßig auszuführenden Phasen des Position-Code-Algorithmus sind:

[2] Im Folgenden wird die ursprüngliche Variante nach Oldenburg und Nilsson als *linearer* Position-Code-Algorithmus bezeichnet.

Aktualisieren	Neuberechnung des Codes zu einem Knoten des Netzes
Sortieren	Sortieren der Knoten bzgl. ihrer Codes
Suchen	Suchen der Intervalle innerhalb der Knotenliste, die den Kontaktbereich eines Segmentes überdecken
Testen	Testen aller Knoten innerhalb der gefundenen Intervalle, ob sie im Halo des Segmentes liegen

Es zeigt sich, dass die Wahl der Zellengröße kritisch für die Laufzeit des Algorithmus ist: Erstreckt sich ein Segment über viele Zellen in vertikaler Richtung sind mehrere Suchoperationen zwischen den Knotenintervallen notwendig. Liegen sehr viele Knoten innerhalb jeder Zelle, ergibt sich innerhalb der Zellen wiederum ein quadratischer Aufwand, da in diesem Gebiet jeder Knoten zu jedem Segment getestet werden muss. Insbesondere in Netzen mit sehr unterschiedlichem Diskretisierungsgrad muss ein Kompromiss gefunden werden, der versucht, beide Extreme zu vermeiden.

4.3.2 Position-Codes auf Basis der Lebesgue-Kurve

Die oben geschilderte Problematik zur Wahl der Zellengröße hängt im Wesentlichen mit der zeilenweisen Nummerierung und der damit implizierten Priorisierung der Achsen bei der Sortierung zusammen. Um sich von dieser Priorisierung zu lösen, kann die Nummerierung über eine raumfüllende Kurve durchgeführt werden. Im Gegensatz zu der Partitionierung mittels raumfüllender Kurven ist in diesem Falle nicht die *Indizierung* (streng monotone Ordnung) bzgl. der Kurve sondern eine *Codierung* gefordert.

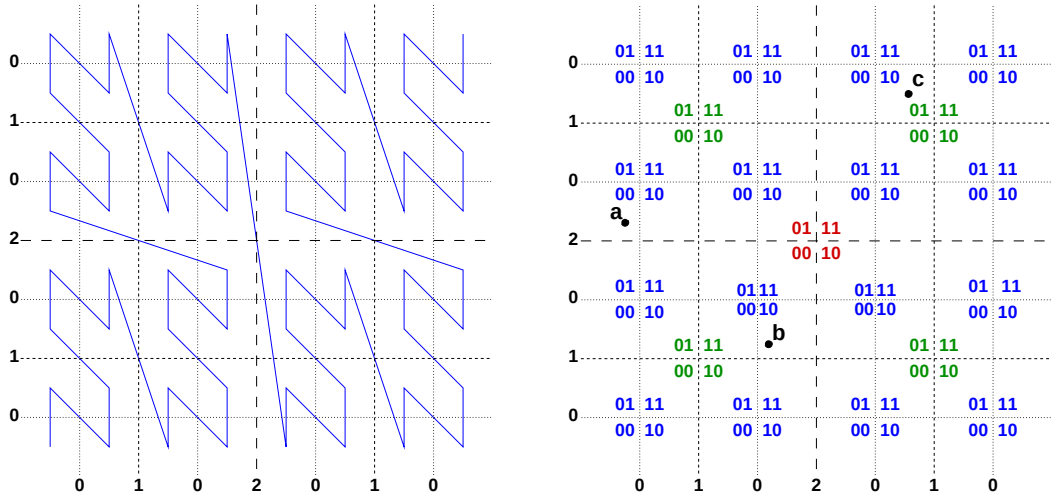


Abbildung 4.7: Lebesgue-Kurve in 8×8 Zellen und abgeleitete Codierung der Zellen

In Abbildung 4.7 sind die Lebesgue-Kurve für die Aufteilung in 8×8 Zellen (links) und die zugehörige Codierung in Binärdarstellung (rechts) angegeben. Die roten Codes sind die höchstwertigen der Separationslinien vom Level zwei, die blauen sind die niederwertigsten der Level null Separation. Tabelle 4.3 zeigt die Codes der in der Abbildung

dargestellten Knoten a, b und c, dezimal und binär. Zudem sind in der Tabelle die zugehörigen Werte sep_x und sep_y angegeben, die für die effiziente Aktualisierung der Codes notwendig sind. sep_x ist die Anzahl der Separationslinien, die links des Knotens liegen, sep_y die Anzahl derer, die unterhalb des Knotens liegen.

Knoten	*code*	sep_x	sep_y
a	$16 \equiv 010000$	$0 \equiv 000000$	$4 \equiv 000100$
b	$14 \equiv 001110$	$3 \equiv 000011$	$2 \equiv 000010$
c	$54 \equiv 110110$	$5 \equiv 000101$	$6 \equiv 000110$

Tabelle 4.3: Knoten im Suchraum mit zugehörigen Werten

Für die Algorithmen zum Initialisieren und Aktualisieren der Position-Codes und zum Testen der Halo-Kandidaten werden die folgenden Daten angenommen: Der Suchraum ist in $nbox \times nbox$ Zellen der Größe $boxsize \times boxsize$ aufgeteilt. Dabei gilt $nbox = 2^{l_{\max}+1}$, d. h. es gibt Separationslinien vom Level 0 bis Level $l_{\max}$. Die Nummerierung der Zellen erfolgt von 0 bis $nbox^2 - 1$. Die Koordinaten eines Knotens a sind gegeben als a_x und a_y. $origin_x$ und $origin_y$ geben die Koordinaten der linken unteren Ecke des Suchraumes an. Dies ermöglicht eine virtuelle Verschiebung des Ursprungs, so dass die Werte $a_x - origin_x$ und $a_y - origin_y$ immer im positiven Bereich liegen. Es ergibt sich eine Binärdarstellung der Länge $2 \cdot (l_{\max} + 1)$. Die eingesetzten Operatoren sind entsprechend Tabelle 4.4 definiert.

Operator	Erläuterung
$\circ$	Konkatenation zweier Bitstrings
$\ll$	Bitstring nach links „shiften“
$\wedge$	Und-Verknüpfung zweier Bitstrings
$\vee$	Oder-Verknüpfung zweier Bitstrings
$\otimes$	exklusive Oder-Verknüpfung zweier Bitstrings

Tabelle 4.4: Binäre Operatoren

4.3.2.1 Initialisieren der Codes

Die Rekursionstiefe der raumfüllenden Kurve ist vorab durch die gewünschte Zellengröße festgelegt, während bei der Partitionierung die Rekursion fortgeführt wird, bis alle Knoten separiert sind. Insbesondere muss während der rekursiven Aufteilung des Gebietes in Zellen der geforderte Code bestimmt werden. Ein Algorithmus zur Berechnung des Codes eines Knotens und der zugehörigen Separatoren-Zähler ist in Abbildung 4.8 angegeben.

Die Berechnung des Codes erfolgt über die Binärdarstellung (vgl. Abb. 4.7). Für jeden Level i eines Separators wird getestet, ob der betrachtete Knoten rechts oder links bzw. über oder unter dem jeweiligen Separator liegt. Je nach Lage wird die letzte Stelle mit eins oder null belegt und der Separatorzähler um die entsprechende Anzahl an Zellen

$code(a) = 0$
$sep_x(a) = 0$
$sep_y(a) = 0$
for $i = l_{\max}$ to 0 **do**
  $code(a) \ll= 1$
  if $(a_x - origin_x) > 2^i \cdot boxsize$ **then**
    $code(a)$ += 1
    $sep_x(a)$ += 2^i
    a_x −= $2^i \cdot boxsize$
  end if
  $code(a) \ll= 1$
  if $(a_y - origin_y) > 2^i \cdot boxsize$ **then**
    $code(a)$ += 1
    $sep_y(a)$ += 2^i
    a_y −= $2^i \cdot boxsize$
  end if
end for

Abbildung 4.8: Initialisieren von *code*, sep_x und sep_y zu Knoten a

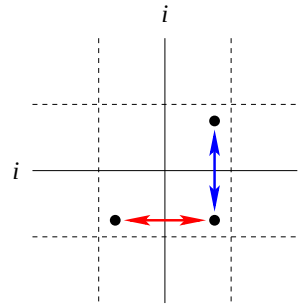
erhöht. Die Position wird ggf. verringert, so dass der verbleibende Suchraum immer genau zwei Separatoren (vertikaler und horizontaler) vom Level i enthält. In jedem Fall wird die Ziffernfolge um eine Stelle nach links verschoben, so dass am Ende die beiden Bits die höchstwertigen Stellen belegen, die den Separatoren des Levels $l_{\max}$ entsprechen.

4.3.2.2 Aktualisieren der Codes

Die regelmäßige Struktur der Lebesgue-Kurve erlaubt eine inkrementelle Aktualisierung des Codes für einen Knoten der in eine andere Zelle gewandert ist. Damit entfällt eine häufige Neuberechnung des Codes mit Aufwand $O(l_{\max})$ je Knoten. In der Analyse zur mittleren Partitionierungsqualität in Abschnitt 3.3.3 ist schon die Tatsache genutzt, dass der Abstand zweier benachbarter Zellen lediglich von dem Level des trennenden Separators abhängig ist. Mit $U_q = \frac{2 \cdot 4^q + 1}{3}$ und $R_p = \frac{4 \cdot 4^p + 2}{3}$ für einen horizontalen Level q bzw. vertikalen Level p Separator können diese Abstände direkt angegeben werden. Da die Codierung in Binärdarstellung in dieser Anwendung intuitiver ist, baut der folgende Algorithmus auf logischen Operationen über Bitstrings auf.

Abbildung 4.9 zeigt den Algorithmus für das Aktualisieren des Position-Codes eines Knotens. Die ersten beiden Schleifen testen hinsichtlich einer horizontalen Verschiebung, die anderen beiden hinsichtlich einer vertikalen. Da die Anzahl der links des Knotens gelegenen Separatoren bekannt ist, kann mit Hilfe der aktuellen Koordinate überprüft werden, ob eine Bewegung nach links oder rechts über eine oder mehrere Zellengrenzen stattgefunden hat.

Für das Aktualisieren des Codes ist es notwendig, den Level des überschrittenen Se-



while $(a_x - origin_x) \leq sep_x(a) \cdot boxsize$ **do**
  $k = \min\{j \mid sep_x(a) \wedge 2^j \neq 0\}$
  $code(a) \otimes= (10)^{k+1}$
  $sep_x(a)--$
end while
while $(a_x - origin_x) > (sep_x(a) + 1) \cdot boxsize$ **do**
  $k = \min\{j \mid (sep_x(a) + 1) \wedge 2^j \neq 0\}$
  $code(a) \otimes= (10)^{k+1}$
  $sep_x(a)++$
end while
while $(a_y - origin_y) \leq sep_y(a) \cdot boxsize$ **do**
  $k = \min\{j \mid sep_y(a) \wedge 2^j \neq 0\}$
  $code(a) \otimes= (01)^{k+1}$
  $sep_y(a)--$
end while
while $(a_y - origin_y) > (sep_y(a) + 1) \cdot boxsize$ **do**
  $k = \min\{j \mid (sep_y(a) + 1) \wedge 2^j \neq 0\}$
  $code(a) \otimes= (01)^{k+1}$
  $sep_y(a)++$
end while

Abbildung 4.9: Aktualisieren von *code*, sep_x und sep_y zu Knoten a

parators zu kennen. Auch dafür ist der Wert von sep_x hilfreich, da die Position des niederwertigsten auf eins gesetzten Wertes genau den gesuchten Level angibt[3]. Ist ein Separator vom Level k überschritten worden, müssen $k+1$ der hinteren Bits umgeschaltet werden, welche die Position in x-Richtung angeben. $(10)^{k+1}$ entspricht dem Wert R_p, der Differenz zwischen den Codes der Zellen rechts und links des Separators. Die Aktualisierung nach einer Verschiebung in y-Richtung wird analog durchgeführt. Das Umschalten der Bits erfolgt über die Bitfolge $(01)^{k+1}$ bei der Überschreitung eines Separators vom Level k.

Um in der Liste der Knoten schnell die Halo-Kandidaten finden zu können, muss die Sortierung bzgl. der Codes ständig erhalten bleiben. Der Grad der Vorsortierung ist im Allgemeinen sehr hoch, da die Verschiebung der Knoten gering und die räumliche Lokalität in der Kurve hoch ist. Um die gewünschte Effizienz der Kontaktsuche zu erzielen, ist ein entsprechender Sortieralgorithmus zu wählen, der eine Vorsortierung der Liste berücksichtigt. An dieser Stelle wird die Liste in einem balancierten Binärbaum gehalten. Die Kosten einer Verschiebeoperation eines Blattes sind im Mittel durch den Logarithmus des Abstands zwischen Lösch- und Einfügeposition gegeben [52].

[3]Die Zählung der Positionen im Bitstring beginnt mit null.

4.3.2.3 Testen der Halo-Kandidaten

Wie bei der linearen Variante des Position-Code-Algorithmus müssen alle Knoten gegen den Halo des betrachteten Segmentes getestet werden, deren Codes einer bestimmten Zellenmenge angehören. Während dieses Gebiet bei ersterem aus mehreren Zeilen besteht, setzt es sich im Falle der Lebesgue-Kurve aus quadratischen Gebieten zusammen. Abbildung 4.10 zeigt ein Segment innerhalb der Lebesgue-Kurve der Tiefe drei. Die

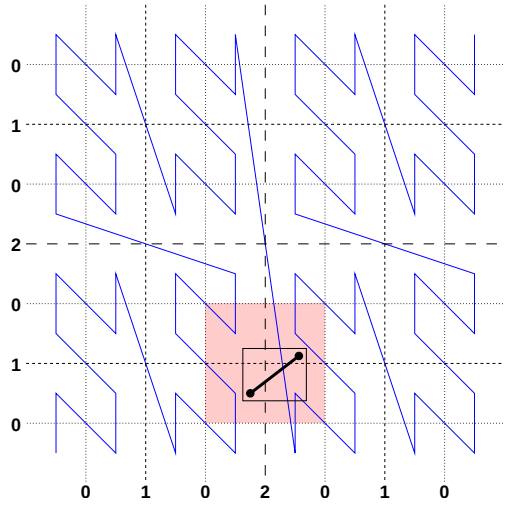


Abbildung 4.10: Suche der Kontaktkandidaten in der Lebesgue-Kurve

Besonderheit bei einer rekursiven raumfüllenden Kurve ist die Tatsache, dass maximal vier Gebiete zu untersuchen sind, und diese den Zellen der entsprechend grobgranulareren Kurve entsprechen. Damit ist die Anzahl der Code-Intervalle, die innerhalb der Gesamtliste aller Knoten gesucht werden müssen unabhängig von der gewählten Zellengröße.

Zunächst wird der Kontaktbereich konstruiert, welcher den Halo des Segmentes enthält. Die Codes zu den Eckpunkten e_1 und e_2, die den Endknoten des Segmentes am nächsten liegen, können wie im Algorithmus in Abbildung 4.9 aus den Codes zu den Knoten gewonnen werden. Die beiden anderen Eckpunkte e_3 und e_4 des Rechtecks ergeben sich durch folgende Bitoperationen:

$$
\begin{aligned}
code(e3) &= code(e1) \otimes \Big((code(e1) \otimes code(e2)) \wedge (10)^{l_{\max}+1} \Big) \\
code(e4) &= code(e2) \otimes \Big((code(e1) \otimes code(e2)) \wedge (10)^{l_{\max}+1} \Big)
\end{aligned}
$$

Durch die äußere exklusive Oder-Verknüpfung werden alle Bits umgeschaltet, durch die sich die Codes der beiden Eckpunkte e_1 und e_2 in ihrer vertikalen Lage unterscheiden.

Im nächsten Schritt werden die bis zu vier Code-Intervalle bestimmt, in denen die Halo-Kandidaten liegen. Dazu muss der Level bestimmt werden, dessen Zelle die Größe des Rechtecks um das Segment übersteigt. Sei *size* die größte Kantenlänge, dann ist der gesuchte Level k gegeben als

$$k = \min\left\{ j \,|\, size < 2^j \cdot boxsize \right\} \ .$$

Anfang und Ende der Code-Intervalle können aus den Codes der Eckpunkte e_j gefunden werden, indem die letzten $2i$ Bits auf 0 bzw. auf 1 gesetzt werden. Dies generiert die Nummern der ersten und letzten Zelle in dem jeweiligen Intervall:

$$\begin{aligned} startcode(e_j) &= code(e_j) \wedge (11)^{l_{\max}+1-i}(00)^k \quad \text{und} \\ stopcode(e_j) &= code(e_j) \vee (00)^{l_{\max}+1-i}(11)^k \quad \text{für } j \in \{1,\ldots,4\} \ . \end{aligned}$$

Jedes der Code-Intervalle $[startcode(e_j), stopcode(e_j)]$ beschreibt ein quadratisches Gebiet der Größe $2^k \cdot boxsize \times 2^k \cdot boxsize$ um einen der Eckpunkte des Rechtecks, welches das untersuchte Oberflächensegment umschließt. Der Level k ist dabei so gewählt, dass das Rechteck von höchstens einem vertikalen und höchstens einem horizontalen Separator vom Level $l \geq i$ geschnitten wird. Damit bilden die bis zu vier quadratischen Teilgebiete ein zusammenhängendes Gebiet, welches das Rechteck vollständig beinhaltet (vgl. Abb. 4.10). Wird das Rechteck von weniger Separatoren von höherem Level geschnitten, sind entweder jeweils zwei oder alle vier Code-Intervalle identisch.

Das Finden der Halo-Kandidaten erfolgt über geeignete Suchoperationen innerhalb der Liste aller Knoten, die gemäß ihrer Position-Codes sortiert ist. Startpunkt der Suche ist jeweils einer der Endknoten des untersuchten Segmentes, da deren Lage innerhalb der Liste bekannt ist und die Code-Intervalle aufgrund der gewählten Codierung über die Lebesgue-Kurve in den meisten Fällen eine räumliche Nähe aufweisen.

Um die gewünschten Effizienzeigenschaften zu erzielen, wird der Suchalgorithmus von der relativen Lage des gesuchten Intervalls zu den beiden bekannten Knoten abhängig gewählt. Ist keiner der beiden Endknoten im gesuchten Intervall enthalten, wird eine exponentielle Suche gestartet, die bei dem Knoten beginnt, dessen Abstand (Differenz der Codes) zum gesuchten Intervall geringer ist. Liegt das Intervall um einen der beiden Endknoten, kann auch eine sequentielle Suche zum Startpunkt des Intervalls ausgeführt werden. Da innerhalb der anschließenden Testphase jeder Knoten im durchlaufenen Bereich untersucht werden muss, wird dabei die gleiche algorithmische Komplexität erreicht.

4.3.3 Erweiterung für die parallele Kontaktsuche

In Kapitel 3 wurde gezeigt, dass sich raumfüllende Kurven für die Partitionierung von FE-Netzen gut eigenen. Daher bietet es sich an, die Nummerierung der Knoten bzgl. ihrer geometrischen Position gleichzeitig auch für die Partitionierung in einer parallelen Kontaktsuchphase einzusetzen. Abgesehen von den Enden eines Intervalls ergeben sich aus raumfüllenden Kurven meist Partitionen die aus größeren rechtwinkligen Gebieten zusammengesetzt sind. Im Gegensatz zu der Methode des *multi-constraint graph*

partitioning (vgl. Abschnitt 4.2.3) kann diese Partitionierung nicht für die allgemeine FE-Berechnung genutzt werden. Vorteilhaft ist jedoch die Tatsache, dass ein homogener Algorithmus eingesetzt wird, dessen Kontaktanfragen im sequentiellen und parallelen vergleichbar ablaufen. Zudem sind die Kosten für die globale Kontaktsuche innerhalb der Partitionen schwer abschätzbar. Eine Partitionierung aufgrund von gewichteten Knoten wird nicht zwingend zu einer gleichmäßigen Verteilung der Rechenlast führen. Die Flexibilität der Partitionierung über eine sortierte Knotenfolge benötigt dahingegen keine vorherige Abschätzung von Rechenlasten, sondern erlaubt die dynamische Annäherung an eine ausgeglichene Lastsituation im Verlauf der Simulation.

Für eine Kontaktanfrage werden wiederum die Code-Intervalle bestimmt, die einen Kontaktbereich eines Segmentes überdecken. Für Anfragen, die über Partitionsgrenzen hinweg verlaufen, da die Intervalle (teilweise) in fremden Partitionen liegen, gibt es zwei Möglichkeiten zur Verarbeitung: Entweder wird die Anfrage vollständig von der Prozesseinheit abgearbeitet, welche die Quelle der Anfrage ist, oder die Anfrage wird in Teilen an die angrenzenden Prozesseinheiten verteilt. Erstere Variante eignet sich insbesondere in parallelen Systemen mit gemeinsamem Speicher. Da nur lesende Operationen in der globalen Position-Code-Liste durchzuführen sind, entstehen keine Zugriffskonflikte. Die zweite Variante ist in Systemen mit verteiltem Speicher sinnvoll, da der Kommunikationsaufwand verringert wird. Es muss nur der Halo des anfragenden Segmentes und das zu untersuchende Teilintervall versandt werden. In der Antwort müssen lediglich die Knoten übermittelt werden, die im Halo liegen. Deren Anzahl wird i. A. deutlich geringer sein, als die der Knoten in den Intervallen, die den Kontaktbereich abdecken.

4.4 Aufwandsanalyse

In diesem Abschnitt werden die Laufzeiten für beide Varianten des Position-Code-Algorithmus analysiert und verglichen. Für die lineare Variante sind von Oldenburg und Nilsson konstante Kosten je Knoten und Segment publiziert [59]. In der Arbeit ist jedoch kein konkretes Analyse-Szenario angegeben, auf dem dieses Ergebnis basiert. Das hier gewählte Szenario zeigt die Schwächen der zeilenweisen Nummerierung, die insbesondere bei einer hohen Varianz im Grad der Diskretisierung auftreten.

Die Analyse bezieht sich auf die Effizienz im mittleren Fall. Für beide Varianten können Fälle konstruiert werden, die zu einer sehr hohen Komplexität führen (vgl. Tab 4.1, S. 70). Für die Lebesgue-Kurve beinhaltet der schlechteste Fall eine Bewegung über den Separator vom höchsten Level oder eine Suchoperation zwischen zwei Gebieten, die vom Separator des höchsten Levels getrennt sind. In beiden Fällen ergibt sich ein Aufwand der vom Grad des höchsten Levels und damit von der Größe des Suchgebietes abhängig ist.

Für die Abschätzung der Laufzeit werden die Vergleichsoperationen zur Durchführung der einzelnen Phasen gezählt. Ihre Anzahl beeinflusst maßgeblich die Laufzeit der Verfahren. Zudem ist das Maß weniger von der tatsächlichen Umsetzung des Algorithmus

in einer Programmiersprache abhängig, als die Anzahl aller Operationen. Die asymptotischen Komplexitäten sind jedoch in beiden Fällen identisch.

4.4.1 Szenario

Um den algorithmischen Aufwand der vorgestellten Variante des Position-Code-Algorithmus abzuschätzen und mit dem linearen zu vergleichen wird zunächst ein Analyse-Szenario definiert (vgl. Abb. 4.11). Es ist eine starke Abstraktion eines tatsächlichen FE-Netzes und erlaubt eine Untersuchung der Algorithmen mit moderatem Aufwand. Dennoch zeigen die Messungen an synthetischen Benchmark-Netzen, dass die hier bestimmten Werte auch auf andere Eingaben übertragbar sind.

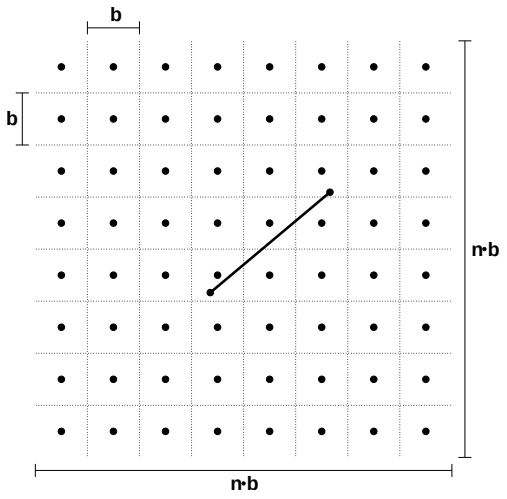


Abbildung 4.11: Analyse-Szenario

Das Gebiet ist in $n \times n$ Zellen der Größe $b \times b$ aufgeteilt. Für die Konstruktion der Lebesgue-Kurve ergibt sich eine Rekursion der Tiefe $\log n$, d. h. die mittleren Separatoren sind vom Level $l_{\max} = \log n - 1$ und $n = 2^{l_{\max}+1}$.[4] Jede Zelle enthält einen Knoten an der Oberfläche des Netzes[5]. Für die Anfrage bzgl. eines Segmentes wird eine beliebige Länge $S = s \cdot b$ und eine beliebige Lage im Raum (Position und Steigung) angenommen.

Die stärksten Abstraktionen ergeben sich in der Verteilung und der Anzahl der Knoten des FE-Netzes. Bei einem gegebenen unstrukturierten FE-Netz ist die Anzahl der Zellen deutlich größer als die der Oberflächenknoten. Zudem ist die Verteilung von der Varianz in der Diskretisierung und dem Verlauf der Oberflächen abhängig. Diese Vereinfachung erlaubt jedoch eine detaillierte Analyse und entbindet von der Frage der optimalen Zellengröße in stark unterschiedlich diskretisierten Netzen. Das Szenario ähnelt dem in Abschnitt 4.2.1 (S. 70) skizzierten. Ein deutlicher Unterschied liegt in der Größe

[4] In diesem Abschnitt wird ausschließlich der Logarithmus zur Basis 2 verwendet. Aus Gründen der Übersichtlichkeit wird auf die explizite Angabe der Basis verzichtet.

[5] Alle inneren Knoten des Netzes sind für die Kontaktbehandlung nicht von Belang und müssen daher in der Analyse nicht berücksichtigt werden.

der Anfrageregion. Das hier vorgestellte Szenario enthält eine variable Größe, während das vorherige eine konstante Größe annimmt, welche aus den anderen dort gegebenen Randbedingungen implizit gegeben ist.

Die Kosten der initialen Bestimmung der Codes aller Oberflächenknoten hat einen sehr geringen Einfluss auf die gesamte Laufzeit der Simulation. Die anderen Phasen der Kontaktbehandlung werden regelmäßig wiederholt und in der expliziten FE-Simulation werden i. A. deutlich mehr Zeitschritte ausgeführt, als Knoten im Netz vorhanden sind. In der linearen Variante können die Codes direkt berechnet werden. Bei der Lebesgue-Kurve sind je Rekursionsschritt drei Vergleichsoperationen durchzuführen: Zwei für die Tests gegen horizontalen und vertikalen Separator vom aktuellen Level und einer für die Schleife über die Rekursionstiefe der Kurve (vgl. Algorithmus in Abb. 4.8). Damit ergeben sich $3 \log n$ Vergleichsoperationen je Knoten.

Da für die Liste der Knoten keine Vorsortierung bzgl. ihres Position-Codes angenommen werden kann, liegen die notwendigen Kosten für die initiale Sortierung bei $O(\log n)$ je Knoten und sind damit von der gewählten Variante unabhängig.

4.4.2 Aktualisieren der Position-Codes

Für die Analyse der Code-Aktualisierung sind zwei Kennzahlen zu bestimmen: Der Aufwand für die Berechnung des Codes und für die Neusortierung der Knotenliste bzgl. ihrer Codes. Im Falle des linearen Position-Codes können die Codes direkt aus den Koordinaten bestimmt werden. Damit ergeben sich Kosten in Höhe einer Vergleichsoperation je Knoten, die für die Abbruchbedingung der Schleife über alle Knoten erforderlich ist.

Bei der Verwendung der Lebesgue-Kurve ergibt sich die inkrementelle Aktualisierung des Codes mit Hilfe des Algorithmus in Abbildung 4.9 (S. 79). Für den Test, ob ein Knoten seit der letzten globalen Kontaktsuche innerhalb der selben Zelle verblieben ist, sind 4 Vergleichsoperationen notwendig (einmaliger Test der Abbruchbedingung für jede while-Schleife). Zusammen mit dem Vergleich in der Schleife über alle Oberflächenknoten ergibt sich ein minimaler Aufwand von 5 Vergleichen je Knoten. Hat ein Knoten einen Separator überschritten, muss dessen Level i bestimmt werden ($k = \min\{j | sep_x(a) \wedge 2^j \neq 0\}$). Dabei werden $t(k) = k + 1$ Vergleiche ausgeführt. Gemittelt über die Häufigkeitsverteilung der Level ergibt sich ein Aufwand von

$$\bar{t} = \sum_{k=0}^{l_{\max}} p(k) \cdot t(k) \leq \sum_{k=0}^{\infty} p(k) \cdot t(k) = \sum_{k=0}^{\infty} \frac{k+1}{2^{k+1}} = \sum_{k=1}^{\infty} \frac{k}{2^k} = 2$$

Vergleichen je überschrittenem Separator. Sei m die Anzahl der überschrittenen Separatoren. Der Aufwand für die Aktualisierung des Codes eines Knotens ergibt sich dann zu

$$T^{\text{Lebesgue}}_{\text{Aktualisieren}} = 5 + 2m \ . \tag{4.1}$$

Ist die Verschiebung der Knoten beschränkt und wird nur ein Separator je Orientierung überschritten, ist der Aufwand durch 9 Vergleichsoperationen beschränkt.

Für die Speicherung der Knotenliste wird ein balancierter Binärbaum eingesetzt. Für die Kosten einer Verschiebe-Operation[6] mit Abstand d kann mit mittleren Kosten $t_v(d)$ in Höhe von

$$t_v(d) = 8\,(\log d + 2) \tag{4.2}$$

durchgeführt werden [52]. Um den Aufwand für die Sortierung der Position-Code-Liste zu bestimmen, muss für beide Varianten über die erwarteten Distanzen d bei den Verschiebeoperationen und den entsprechenden Kosten $t_v(d)$ gemittelt werden.

Für den linearen Position-Code ergeben sich offensichtlich Abstände, die von der Verschiebungsrichtung der betroffenen Knoten abhängig sind. Im gegebenen Szenario führt der Übergang über einen vertikalen Separator zu einer Verschiebung um eine Position mit Kosten $t_v(1)$. Ein Übergang über einen horizontalen Separator resultiert in einer Distanz von n Stellen und damit Kosten in Höhe von $t_v(n)$. Zwei Operationen sind notwendig, um zu prüfen, ob der Knoten an der korrekten Position der Liste steht. Die gesamten Kosten für m_1 vertikale und m_2 horizontale Übergänge ergeben sich zu

$$\begin{aligned} T^{\text{linear}}_{\text{Sortieren}} &= 2 + m_1 \cdot t_v(1) + m_2 \cdot t_v(n) \\ &= 2 + m_1 \cdot 8\,(1+2) + m_2 \cdot 8\,(\log n + 2) \\ &= 2 + m_1 \cdot 24 + m_2 \cdot (8 \log n + 16)\ . \end{aligned} \tag{4.3}$$

Bei einer beschränkten Verschiebung der Knoten um maximal eine Zelle ergeben sich Kosten von weniger als $42 + 8 \cdot \log n$ Vergleichsoperationen.

Innerhalb der Lebesgue-Kurve sind die Distanzen nicht nur von der Bewegungsrichtung, sondern vielmehr von dem Level des überschrittenen Separators abhängig. Die Abstände sind $U_q = \frac{2 \cdot 4^q + 1}{3}$ für einen horizontalen Separator vom Level q und $R_p = \frac{4 \cdot 4^p + 2}{3}$ für einen vertikalen Separator vom Level p (vgl. Lemma 2.1, S. 10). Gemittelt über die Wahrscheinlichkeiten der verschiedenen Level ergeben sich die Kosten als

$$\begin{aligned} T^{\text{Lebesgue}}_{\text{Sortieren}} &= 2 + m_1 \sum_{k=0}^{l_{\max}} p(k) \cdot t_v(R_k) + m_2 \sum_{k=0}^{l_{\max}} p(k) \cdot t_v(U_k) \\ &= 2 + m_1 \sum_{k=0}^{l_{\max}} \frac{1}{2^{k+1}} \cdot 8 \cdot \left[\log\left(\frac{4 \cdot 4^k + 2}{3} \right) + 2 \right] \\ &\quad + m_2 \sum_{k=0}^{l_{\max}} \frac{1}{2^{k+1}} \cdot 8 \cdot \left[\log\left(\frac{2 \cdot 4^k + 1}{3} \right) + 2 \right] \\ &\leq 2 + 8 m_1 \sum_{k=0}^{l_{\max}} \frac{1}{2^{k+1}} \cdot [2k + 4] \\ &\quad + 8 m_2 \sum_{k=0}^{l_{\max}} \frac{1}{2^{k+1}} \cdot [2k + 2] \\ &= 2 + 8 m_1 \sum_{k=0}^{l_{\max}} \frac{k}{2^k} + \frac{2}{2^k} \end{aligned}$$

[6] Entfernen eines Schlüssels und anschließendes Einfügen an einer Stelle mit Abstand d zur alten Position.

$$
\begin{aligned}
& +8m_2 \sum_{k=0}^{l_{\max}} \frac{k}{2^k} + \frac{1}{2^k} \\
\leq \quad & 2 + 48m_1 + 32m_2 \ . \qquad (4.4)
\end{aligned}
$$

Gilt $m_1, m_2 \leq 1$ sind im Mittel höchstens 82 Vergleiche notwendig, um einen Knoten innerhalb der Liste der Oberflächenknoten bzgl. seines veränderten Position-Codes zu verschieben.

4.4.3 Testen der Halo-Kandidaten

Im Szenario in Abbildung 4.11 ist die Halo-Anfrage eines Segmentes mit beliebiger Lage und Länge vorgesehen. Für die Analyse der beiden Varianten des Position-Code-Algorithmus werden die Höhe, die Breite, die Fläche und die maximale Ausdehnung des resultierenden Kontaktbereiches im mittleren Fall benötigt. Sei die Länge des Ober-

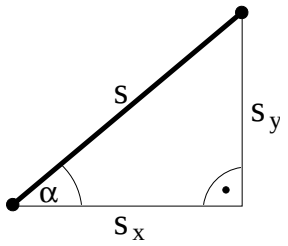


Abbildung 4.12: Kontaktsegment im Suchraum

flächensegmentes $s = l \cdot b$ als Vielfaches der Größe einer Zelle gegeben. Sei α der Winkel, der die Steigung festlegt (vgl. Abb. 4.12). Aus Gründen der Symmetrie ist es ausreichend, α als gleichverteilt in dem Intervall $[0, \pi/2]$ anzunehmen. Die mittlere Breite $\bar{s}_x$ und die mittlere Höhe $\bar{s}_y$ betragen

$$
\begin{aligned}
\bar{s}_x &= \frac{1}{\pi/2} \int_0^{\pi/2} s \cdot \cos\alpha \, d\alpha = \frac{1}{\pi/2} \Big[s \cdot \sin\alpha \Big]_0^{\pi/2} = \frac{2s}{\pi} \quad \text{und} \\
\bar{s}_y &= \frac{1}{\pi/2} \int_0^{\pi/2} s \cdot \sin\alpha \, d\alpha = \frac{1}{\pi/2} \Big[-s \cdot \cos\alpha \Big]_0^{\pi/2} = \frac{2s}{\pi} \ .
\end{aligned}
$$

Im mittleren Fall beträgt das Maximum $\bar{s}_{\max}$ aus Breite s_x und Höhe s_y

$$
\begin{aligned}
\bar{s}_{\max} &= \frac{1}{\pi/2} \int_0^{\pi/2} \max(s \cdot \cos\alpha, s \cdot \sin\alpha) \, d\alpha \\
&= \frac{1}{\pi/2} \left(\int_0^{\pi/4} \max(s \cdot \cos\alpha, s \cdot \sin\alpha) \, d\alpha + \int_{\pi/4}^{\pi/2} \max(s \cdot \cos\alpha, s \cdot \sin\alpha) \, d\alpha \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{\pi/2}\left(\int_0^{\pi/4} s\cdot\cos\alpha\,d\alpha + \int_{\pi/4}^{\pi/2} s\cdot\sin\alpha\,d\alpha\right)\\
&= \frac{1}{\pi/2}\left(\frac{\sqrt{2}}{2}s + \frac{\sqrt{2}}{2}s\right)\\
&= \frac{2\sqrt{2}}{\pi}s\ .
\end{aligned}
$$

Die mittlere Fläche $\bar{s}_f$ eines Rechtecks $s_y \times s_x$ beträgt

$$
\begin{aligned}
\bar{s}_f &= \frac{1}{\pi/2}\int_0^{\pi/2} s\cdot\cos\alpha\cdot s\cdot\sin\alpha\,d\alpha\\
&= \frac{1}{\pi/2}\left[\frac{1}{2}s^2\cdot\sin^2\alpha\right]_0^{\pi/2}\\
&= \frac{s^2}{\pi}\ .
\end{aligned}
$$

Für den Test gegen den Halo müssen zunächst die entsprechenden Intervalle gesucht werden, die den Kontaktbereich überdecken. Für diese Phase der Kontaktsuche sind die innerhalb des Binärbaumes sortierten Listen in Felder mit direktem Zugriff kopiert. Dieses verringert zwar nicht die Komplexität der Suchoperationen, die in beiden Fällen $O(\log n)$ beträgt. Die Konstanten sind jedoch geringer und die Algorithmen sind einfacher umzusetzen. Für die binäre Suche zwischen zwei Positionen n_1 und n_2, $n_1 < n_2$ ist die Anzahl der Vergleiche auf $2\log(n_2 - n_1) + 2$ beschränkt. Die exponentielle Suche mit Startpunkt n_1 und gesuchter Position n_2 erfordert mit $4\log(n_2 - n_1) + 4$ doppelt so viele Operationen.

Bei der linearen Variante des Position-Code-Algorithmus erstreckt sich der Anfragebereich im Mittel über $\lceil\bar{s}_y\rceil + 1$ Zeilen des gesamten Suchraumes. Jede dieser Zeilen ist $\lceil\bar{s}_x\rceil + 1$ Zellen lang (vgl. Abb. 4.6, S. 75). Um den ersten Halo-Kandidaten innerhalb einer Zeile zu finden, wird jeweils eine exponentielle Suche durchgeführt. Die erste wird von einem der beiden Endknoten des betrachteten Segmentes gestartet, alle weiteren vom letzten betrachteten Halo-Kandidaten. Damit ergeben sich $\lceil\bar{s}_y\rceil + 1$ Suchanfragen mit Aufwand $4\log(n) + 4$ und einem weiteren Vergleich für die Abbruchbedingung der Schleife über alle Suchintervalle. Die Gesamtkosten ergeben

$$
\begin{aligned}
T^{\text{linear}}_{\text{Suchen}} &= (\lceil\bar{s}_y\rceil + 1)\cdot(4\log n + 5)\\
&\leq \left(\frac{2s}{\pi} + 2\right)\cdot(4\log n + 5)\\
&= 10 + \frac{10s}{\pi} + 8\left(\frac{s}{\pi} + 1\right)\cdot\log n\ . \qquad (4.5)
\end{aligned}
$$

Damit existiert neben einer Konstanten ein Term, der von der Länge des Segmentes abhängig ist und ein weiterer, der zusätzlich von der Größe des gesamten Suchraumes abhängig ist.

Für jeden der Knoten im Kontaktbereich des Segmentes muss getestet werden, ob er im Halo liegt. Dieser Test benötigt drei Vergleiche plus einen weiteren für die Schleife über die Knoten. Die Anzahl der Knoten im Kontaktbereich ist durch die Anzahl der überdeckten Zellen gegeben. Sie kann durch die mittlere Fläche ($\bar{s}_f$) plus zwei weitere Zeilen und zwei weitere Spalten mittlerer Länge ($\bar{s}_x, \bar{s}_y$) abgeschätzt werden. Der Test der Halo-Kandidaten erfordert damit einen Aufwand von

$$\begin{aligned} T_{\text{Testen}}^{\text{linear}} &= (3+1)\left[\bar{s}_f + 2\left(\lceil \bar{s}_y \rceil + \lceil \bar{s}_x \rceil + 2\right)\right] \\ &\leq 4\left[\frac{s(s+8)}{\pi} + 8\right] \\ &= 32 + \frac{4s^2 + 32s}{\pi} \end{aligned} \tag{4.6}$$

Vergleichen.

Die Analyse für die Suchkosten beim Position-Code auf Basis der Lebesgue-Kurve erfolgt anhand des Algorithmus in Abschnitt 4.3.2.3 (S. 80). Die Bestimmung des Levels k, dessen Größe den Kontaktbereich übersteigt, erfordert $k+1$ Vergleiche und ist von der Größe des Segmentes abhängig. Damit gilt

$$k = \lceil \log\left(\lceil \bar{s}_{\max} \rceil\right) \rceil = \left\lceil \log\left(\left\lceil \frac{2\sqrt{2}}{\pi} s \right\rceil\right)\right\rceil \leq \log\left(\frac{2\sqrt{2}}{\pi} s\right) + 1 < \log s + 1 \ . \tag{4.7}$$

Für die Berechnung und die Sortierung der Codes aller Eckpunkte werden weitere sieben Vergleiche ausgeführt. Um die bis zu vier Intervalle des Kontaktbereichs zu suchen, finden vier Suchvorgänge statt: Mindestens einer dieser Suchvorgänge findet innerhalb eines Intervalls statt und ist damit lediglich von der Fläche des Ziellevels (4^k) abhängig, so dass

$$t_{\text{lokal}}(k) \leq 4 \log(4^k) + 4 = 8k + 4 \ . \tag{4.8}$$

Das Finden der (bis zu) drei anderen Intervalle des Kontaktbereichs hängt von dem Level der Separatoren ab, die den Kontaktbereich schneiden (vgl. Abb. 4.10, S. 80). Sei j der Level des Separators, der das zu suchende Intervall vom aktuellen trennt. Beginnt die Suche bei dem letzten Knoten des aktuellen Intervalls, so ist die Distanz, über welche die Suche ausgeführt wird, durch $U_j = \frac{2 \cdot 4^j + 1}{3}$ (horizontaler Separator) und $R_j = \frac{4 \cdot 4^j + 2}{3}$ (vertikaler Separator) begrenzt (vgl. Lemma 2.1, S. 10). Für einen horizontalen Separator ergibt sich über die Wahrscheinlichkeitsverteilung der Level $j > k$ ein Aufwand von

$$\begin{aligned} t_{\text{global}}^{\text{h}}(k) &\leq \sum_{j=k}^{l_{\max}} p(j) \cdot t(j) \\ &= \sum_{j=k}^{l_{\max}} \frac{1}{2^{j-k+1}} \cdot 4 \log\left(U_j\right) + 4 \\ &\leq \sum_{j=k}^{l_{\max}} \frac{1}{2^{j-k+1}} \cdot 4\left(2j\right) + 4 \end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=0}^{l_{\max}-k} \frac{1}{2^{j+1}} \cdot 4\,(2(j+k)+1) \\
&\leq \sum_{j=0}^{\infty} \frac{2}{2^{j}} \cdot (2(j+k)+1) \\
&= \sum_{j=0}^{\infty} \frac{4j}{2^{j}} + \sum_{j=0}^{\infty} \frac{4k}{2^{j}} + \sum_{j=0}^{\infty} \frac{1}{2^{j}} \\
&= 8 + 8k + 2 \\
&= 8k + 10 \qquad (4.9)
\end{aligned}
$$

Vergleichen. An einem vertikalen Separator ergibt sich nach analoger Überlegung ein Aufwand von

$$
\begin{aligned}
t^{\mathrm{v}}_{\mathrm{global}}(k) &\leq \sum_{j=k}^{l_{\max}} p(j) \cdot t(j) \\
&= \sum_{j=k}^{l_{\max}} \frac{1}{2^{j-k+1}} \cdot 4 \log\left(R_j\right) + 4 \\
&\leq \sum_{j=k}^{l_{\max}} \frac{1}{2^{j-k+1}} \cdot 4\,(2j+1) + 4 \\
&\leq 8k + 12 \qquad (4.10)
\end{aligned}
$$

Vergleichen.

Werden zwei der entfernten Suchoperationen über den horizontalen Separator und eine über den vertikalen Separator ausgeführt, ergeben sich aus (4.8), (4.9) und (4.10) Gesamtkosten von

$$
\begin{aligned}
T^{\mathrm{Lebesgue}}_{\mathrm{Suchen}} &\leq k + 1 + 7 + t_{\mathrm{lokal}}(k) + 2t^{\mathrm{h}}_{\mathrm{global}}(k) + t^{\mathrm{v}}_{\mathrm{global}}(k) \\
&\leq k + 8 + 8k + 4 + 2\,(8k + 10) + 8k + 12 \\
&= 44 + 33k \\
&< 44 + 33\,(\log s + 1) \qquad (4.7) \\
&= 77 + 33 \log s \qquad (4.11)
\end{aligned}
$$

Vergleichsoperationen. Diese verringern sich um bis zu $56 + 24 \log s$ Operationen, falls der Kontaktbereich durch weniger als vier Intervalle vom Level k abgedeckt wird oder sich die beiden Endknoten in unterschiedlichen Intervallen befinden. In beiden Fällen verringert sich die Anzahl der Suchoperationen außerhalb des Bereiches der Größe 4^k. Wichtig ist, dass die Anzahl der ausgeführten Vergleichsoperationen im Mittel von der Größe des gesamten Suchraums unabhängig ist. Dies ergibt sich aus der Tatsache, dass die Anzahl der Intervalle auf höchstens vier beschränkt ist und die lokalitätserhaltenden Eigenschaften der raumfüllenden Kurve im Mittel eine begrenzte Distanz zwischen diesen Intervallen garantieren.

Die Suchphase kann optimiert werden, in dem die Folge der Suchoperationen und die jeweilige Startposition geschickt gewählt wird. Dafür ist zu gewährleisten, dass zwei

der entfernten Suchen über den Separator (horizontal oder vertikal) verlaufen, der von kleinerem Level ist und höchstens eine über den von höherem Level. Damit ergibt sich eine Abhängigkeit bei der Wahrscheinlichkeitsverteilung, deren Berücksichtigung in den Berechnungen (4.9) und (4.10) zu einer Verringerung der Gesamtkosten führt. Die Komplexität mit der Abhängigkeit von der Segmentlänge ($\log s$) bleibt jedoch erhalten.

Wie beim linearen Position-Code muss für jeden Knoten innerhalb des Kontaktbereiches getestet werden, ob er im Halo des Oberflächensegmentes liegt. Jedes der bis zu vier Intervalle umfasst 4^k Zellen und damit im gegebenen Szenario 4^k Knoten. Für jeden Knoten sind bis zu drei Vergleiche notwendig, um festzustellen, ob er innerhalb oder außerhalb des Halos liegt. Ein weiterer ist für die Abbruchbedingung der Schleife über alle Knoten erforderlich. Daraus folgt ein Gesamtaufwand von bis zu

$$\begin{aligned} T_{\text{Testen}}^{\text{Lebesgue}} &\leq (3+1) \cdot 4 \cdot 4^k \\ &< 16 \cdot 4^{\log s + 1} \\ &= 64 s^2 \end{aligned} \tag{4.7} \tag{4.12}$$

Vergleichen.

4.4.4 Zusammenfassung

Die Ergebnisse des vorherigen Abschnitts sind in Tabelle 4.5 zusammengefasst. Für die Phasen Aktualisieren und Sortieren sind die minimalen Kosten hervorgehoben. Die anderen Vergleichsoperationen fallen nur dann an, wenn sich ein Knoten in eine andere benachbarte Zelle des virtuellen Position-Code-Gitters bewegt hat. In den Termen der Spalte zur Lebesgue-Kurve sind die Konstanten größer als bei der linearen Position-Code-Variante. Diese Tatsache wird in Abschnitt 5.3 durch Messungen an Benchmark-Netzen bestätigt, die innerhalb einer Applikation durchgeführt werden. Allerdings ergeben sich in den Phasen Sortieren und Suchen bei der linearen Variante Abhängigkeiten von der Anzahl aller Zellen, die sich aus der Größe der Eingabeinstanz ergibt. Desweiteren ist die Suchphase im Falle der Lebesgue-Kurve lediglich vom Logarithmus der Segmentlänge abhängig. Dieser Unterschied zeigt sich insbesondere dann, wenn es eine große Varianz im Diskretisierungsgrad des Netzes gibt. Hier zeigt sich, dass der Einfluss der Zellengröße auf die Effizienz des Position-Code-Algorithmus beim Einsatz der Lebesgue-Kurve deutlich geringer ist.

Programmphase	linear	Lebesgue
Aktualisieren	$\mathbf{1}$	$\mathbf{5} + 2$
Sortieren	$\mathbf{2} + 40 + 8 \log n$	$\mathbf{2} + 80$
Suchen	$10 + \frac{10s}{\pi} + 8\left(\frac{s}{\pi} + 1\right) \log n$	$77 + 33 \log s$
Testen	$32 + \frac{4s^2 + 32s}{\pi}$	$64 s^2$

Tabelle 4.5: Gegenüberstellung der Vergleichsoperationen beider Varianten des Position-Code-Algorithmus

Das Quadrat der Segmentlänge ist maßgebend für die Laufzeit beim Test der Halo-Kandidaten, sowohl für die lineare Nummerierung, als auch für die Lebesgue-Kurve. Diese Terme zeigen die Problematik der notwendigen Abstraktionen, die in dem gegebenen Szenario gemacht wurden. Die Dichte der Oberflächenknoten ist von der Länge des untersuchten Oberflächensegmentes unabhängig gewählt, so dass die Anzahl der Knoten im Kontaktbereich eines Segmentes quadratisch zu dessen Länge wächst. Um die numerische Stabilität und damit eine befriedigende Abbildung der physikalischen Vorgänge zu erreichen, muss die Diskretisierung in Bereichen von Kontaktsituationen jedoch auf beiden Seiten von vergleichbarem Grad sein. Daher wird die Anzahl der Knoten im Allgemeinen im Kontaktbereich eines Segmentes konstant sein.

4.5 Logarithmische Index-Distanz

Der vorherige Abschnitt hat gezeigt, welchen Einfluss die Wahl der Nummerierung auf die Laufzeit des Position-Code-Algorithmus hat. Für eine Codierung gemäß der Lebesgue-Kurve ergeben sich asymptotisch geringere Kosten als bei der zeilenweisen Nummerierung, die ursprünglich vorgeschlagen wurde. Aus diesem Ergebnis ergibt sich die Frage: Sind andere raumfüllenden Kurven für diese Aufgabe besser geeignet?

Bei der Betrachtung der Phasen Aktualisieren und Testen ergeben sich offensichtliche Antworten: Das Testen der Halo-Kandidaten ist für alle rekursiven raumfüllenden Kurven identisch, da lediglich die Form der Aufteilung und nicht die Nummerierung selbst in die Analyse einfließt. Für das Aktualisieren der Codes hat die Lebesgue-Kurve den Vorteil der regelmäßigen Struktur. Für andere Kurven, wie Hilbert- und $\beta\Omega$-Kurve, ist die Kenntnis über den Level eines überschrittenen Separators nicht ausreichend. Vielmehr muss die Position innerhalb der Basisstruktur bekannt sein, in die der überschrittene Separator eingefügt wurde. Für jeden Knoten müssen bei der initialen Code-Berechnung die $l_{\max}$ Basisstrukturen gespeichert werden, in denen er liegt. Bei einer Bewegung in eine benachbarte Gitterzelle über einen Level j kann die Neuberechnung bei der $(l_{\max} - j)$-ten Basisstruktur neu aufgesetzt werden. Im Mittel ergeben sich auch in diesem Fall konstante Kosten.

Ein Vergleich der Analysen für die Phasen Sortieren und Suchen zeigt, dass in beiden Fällen nur der mittlere Abstand (über die Codierung) zu einer geometrisch benachbarten Zelle von der Art der gewählten Kurve abhängig ist (vgl. (4.4) bzw. (4.9) und (4.10)). An diesen Stellen ergeben sich jeweils Terme der Form[7]

$$\sum_{k=0}^{l_{\max}} p(k) \cdot \log(R_k) \qquad \text{oder} \qquad \sum_{k=0}^{l_{\max}} p(k) \cdot \log(U_k) \ ,$$

wobei R_k und U_k die Index-Distanzen zwischen zwei, durch einen Separator vom Level k getrennte Zellen sind. Die Definition der *logarithmischen Index-Distanz* ist eine Verallgemeinerung dieser Terme für beliebige rekursive raumfüllende Kurven. Über dieses

[7] In diesem Abschnitt wird ausschließlich der Logarithmus zur Basis 2 verwendet. Aus Gründen der Übersichtlichkeit wird auf die explizite Angabe der Basis verzichtet.

Maß werden die Hilbert-, die Lebesgue- und die $\beta\Omega$-Kurve hinsichtlich ihrer Eignung für die globale Kontaktsuche im Position-Code-Algorithmus verglichen.

4.5.1 Definition des Qualitätsmaßes

Definition 4.1 (Logarithmische Index-Distanz)
Sei $curve_n : \{1, \ldots, n^2\} \rightarrow \{1, \ldots, n\}^2$ *eine diskrete raumfüllende Kurve (zweidimensionales Indizierungsschema).* $(k, l) = curve_n(i)$ *ist die Position der Zelle* i *im* $n \times n$ *Gitter. Die Index-Distanzen zu den vier direkten Nachbarn von* i *sind gegeben als:*

$$\begin{aligned} d_1^{curve_n}(i) &= |i - curve_n^{-1}(k-1, l)| + 1 \\ d_2^{curve_n}(i) &= |i - curve_n^{-1}(k, l-1)| + 1 \\ d_3^{curve_n}(i) &= |i - curve_n^{-1}(k+1, l)| + 1 \\ d_4^{curve_n}(i) &= |i - curve_n^{-1}(k, l+1)| + 1 \ , \end{aligned}$$

wobei r_j *undefiniert ist, falls der entsprechende Nachbar im* $n \times n$ *Gitter nicht existiert, d. h. die entsprechende Zelle liegt am Rand des* $n \times n$ *Gitters.*

Die Index-Distanz für eine Kurve $curve_n$ *im schlechtesten Fall ergibt sich als:*

$$\mathcal{D}_{\max}^{curve_n} = \max\{d_j^{curve_n}(i) | i \in \mathbb{N} \text{ und } j \in \{1, \ldots, 4\}\} \ .$$

Im mittleren Fall gilt:

$$\mathcal{D}_{\text{avg}}^{curve_n} = \text{avg}\{\log(d_j^{curve_n}(i)) | i \in \mathbb{N} \text{ und } j \in \{1, \ldots, 4\}\} \ ,$$

wobei avg *das arithmetische Mittel über alle definierten Distanzen ist.*

Die Verallgemeinerung zu den Index-Distanzen eines Indizierungsschemas curve sind

$$\begin{aligned} \mathcal{D}_{\max}^{curve} &= \max\{\mathcal{D}_{\max}^{curve_n} | n \in \mathbb{N}\} \quad \text{und} \\ \mathcal{D}_{\text{avg}}^{curve} &= \max\{\mathcal{D}_{\text{avg}}^{curve_n} | n \in \mathbb{N}\} \ . \end{aligned} \tag{4.13}$$

Anmerkung 4.1 Die Definition des mittleren Falls ergibt sich durch die Aufwandsanalyse für die globale Kontaktsuche mittels raumfüllender Kurven in Abschnitt 4.4. In zwei der vier Algorithmen-Phasen sind die Kosten der Kernoperation durch den Logarithmus aus der Index-Distanz zum direkten Nachbarn gegeben. Die Mittelwertanalyse erfolgt über das arithmetische Mittel aller möglichen Positionen im Gitter. Daher beschreibt das arithmetische Mittel über alle *logarithmischen Index-Distanzen* im Gitter den Aufwand dieser Kernoperation im mittleren Fall und erlaubt eine Bewertung verschiedener Indizierungsschemata für den Einsatz in der globalen Kontaktsuche.

4.5.2 Qualität im schlechtesten Fall

In Tabelle 4.6 sind die Index-Distanzen für die untersuchten Indizierungsschemata angegeben. Zum Vergleich ist zusätzlich der Wert bei einer zeilenweisen Nummerierung

Indizierungs-schema	Index-Distanz (max)
Hilbert	$\frac{5N-2}{6}+2$
Lebesgue	$\frac{N+2}{3}$
H-Indizierung	N
$\beta\Omega$-Indizierung	N
zeilenweise	$\sqrt{N}$

Tabelle 4.6: Index-Distanz im schlechtesten Fall ($N = n^2$)

der Zellen eines Gitters angegeben. Die Problemstellung entspricht der Bandbreitenoptimierung innerhalb einer dünnbesetzten Matrix. Ergibt sich deren Struktur aus einem Gitter, ist die zeilenweise Ordnung die optimale Lösung.

H- und $\beta\Omega$-Indizierung sind zirkulär. Damit sind die erste und die letzte Zelle der Kurve benachbart, welche eine Index-Distanz von N haben.

Für die Lebesgue-Kurve können alle Werte exakt bestimmt werden. Die Index-Distanzen an einem vertikalen Separator vom Level i betragen $R_l = \frac{4 \cdot 4^l + 2}{3}$ und an einem horizontalen Separator $U_l = \frac{2 \cdot 4^l + 1}{3}$ (vgl. Lemma 2.1, S. 10). Damit ergibt sich die maximale Distanz bei dem vertikalen Separator vom höchsten Level, $l_{\max} = \log_4(N) - 1$, von

$$\mathcal{D}_{\max}^{\text{Lebesgue}_n} = \frac{N+2}{3} \ .$$

Die maximale Index-Distanz der Hilbert-Kurve ergibt sich am unteren Ende des mittleren vertikalen Separators.[8] Die Distanz umfasst $N/2$ Zellen in der oberen Hälfte des Gitters. Während der Verfeinerung der Kurve, sind die Viertel, die die unteren mittleren Zellen enthalten jeweils von dem gleichen Typ: $\sqsupset$ rechts und $\sqsubset$ links des Separators (vgl. Abb. 2.3, S. 9). Aufgrund dieser Selbstähnlichkeit fällt in jeder Verfeinerungsstufe die Hälfte des jeweiligen Quadranten in die Index-Distanz. Dies ergibt

$$\sum_{l=1}^{\log n} \frac{1}{2} \cdot \frac{1}{4^l} N = \frac{N-1}{6}$$

Zellen für die unteren beiden Quadranten. Zusammen mit den beiden mittleren unteren Zellen folgt die gesamte Index-Distanz

$$\mathcal{D}_{\max}^{\text{Hilbert}} = \frac{N}{2} + 2\frac{N-1}{6} + 2 = \frac{5N-2}{6} + 2 \ ,$$

welche für große Gitter gegen $\frac{5}{6}N$ konvergiert.

[8] Unten bzgl. der in dieser Arbeit gewählten Darstellung (vgl. Abb. 2.3, S. 9).

4.5.3 Qualität im mittleren Fall

Das Verhalten der logarithmischen Index-Distanz im mittleren Fall ist in Abbildung 4.13 dargestellt. In einer Simulationsumgebung sind die mittleren logarithmischen Distanzen für die untersuchten Indizierungsschemata auf Gittern verschiedener Größe bestimmt worden. Die gemessenen Werte deuten an, dass die gesuchten Distanzen in einem engen Intervall zwischen 2.5 und 2.7 liegen.

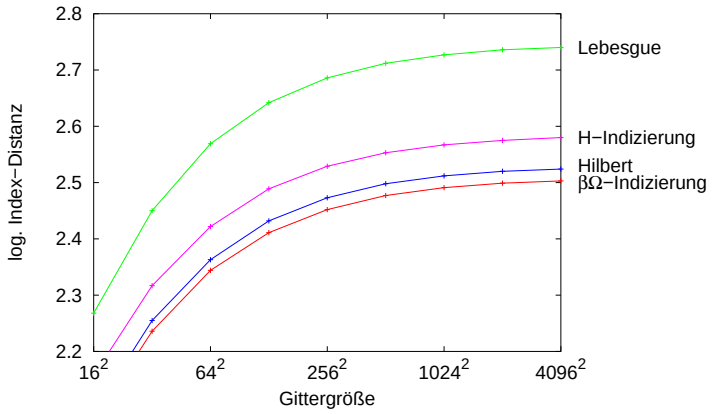


Abbildung 4.13: Logarithmischen Index-Distanz im mittleren Fall

Der Verlauf der Messkurven zeigt, dass die Werte für größere Gitter monoton steigen. Dieses Verhalten kann für die rekursiven Indizierungsschemata auch bewiesen werden (s. u.). Damit sind alle experimentell ermittelten Werte gleichzeitig untere Schranken für die gesuchte mittlere logarithmische Index-Distanz.

Im Folgenden bezeichnet $\mathcal{D}_{\text{avg}}^{curve}(k)$ die mittleren logarithmischen Index-Distanzen an den (vertikalen und horizontalen) Separatoren des k-ten Levels und $\mathcal{D}_{\text{avg}}^{curve}(0,\ldots,k-1) = \mathcal{D}_{\text{avg}}^{curve_{2^k}}$ die mittlere logarithmische Index-Distanz im $2^k \times 2^k$-Gitter, d. h. die mittleren Kosten an den Separatoren 0 bis $k-1$.

Lemma 4.1 *Sei curve eine rekursive raumfüllende Kurve und $k \in \mathbb{N}$. Für die mittlere logarithmisch Index-Distanz gilt*

$$\mathcal{D}_{\text{avg}}^{curve}(0,\ldots,k) > \mathcal{D}_{\text{avg}}^{curve}(0,\ldots,k-1) \ .$$

Beweis: Aufgrund der rekursiven Aufteilung in vier Quadrate sind die Index-Distanzen an den Separatoren vom Level k fast viermal so groß wie an denen vom Level $k+1$ (vgl. Index-Distanzen in den Abbildungen 4.14 und 4.15). Sei $d(k,\xi)$ die Index-Distanz am Separator vom Level k an der relativen Position ξ. Dann gilt für innerhalb einer rekursiven raumfüllenden Kurve aufgrund der Selbstähnlichkeit, dass $d(k+1,\xi) \geq 4d(k,\xi) - 3$. Daraus folgt für die mittlere Distanz an den Separatoren eines Levels:

$$\mathcal{D}_{\text{avg}}^{curve}(k+1) > \mathcal{D}_{\text{avg}}^{curve}(k)$$

Innerhalb eines Gitters der Größe $2^{k+1} \times 2^{k+1}$ gibt es zwei Separatoren vom Level k, einen horizontalen und einen vertikalen. $2 \cdot (2^{k+1} - 2)$ Separatoren sind von einem Level $j < k$. Daher gilt

$$\mathcal{D}_{\text{avg}}^{curve}(0, \ldots, k) = \frac{\mathcal{D}_{\text{avg}}^{curve}(0, \ldots, k-1) \cdot (2^{k+1} - 2) + \mathcal{D}_{\text{avg}}^{curve}(k)}{2^{k+1} - 1}$$

und damit folgt

$$\begin{aligned} & \mathcal{D}_{\text{avg}}^{curve}(k) > \mathcal{D}_{\text{avg}}^{curve}(k-1) \\ \Leftrightarrow \quad & \mathcal{D}_{\text{avg}}^{curve}(0, \ldots, k) > \mathcal{D}_{\text{avg}}^{curve}(0, \ldots, k-1) \ . \end{aligned}$$

□

Die folgenden Untersuchungen beziehen sich auf die logarithmische Index-Distanz in sehr großen Gittern.

4.5.3.1 Lebesgue-Kurve

Aufgrund ihrer regulären Struktur ohne Spiegelungen oder Drehungen bei der rekursiven Konstruktion ist auch für dieses Qualitätsmaß die Lebesgue-Kurve gut zu analysieren.

Satz 4.1 *Für die logarithmische Index-Distanz der Lebesgue-Kurve gilt*

$$2.745 < \mathcal{D}_{\text{avg}}^{\text{Lebesgue}} < 2.746 \ .$$

Beweis: Für jeden Separator ist die Index-Distanz angrenzender Zellen lediglich von der Ausrichtung und dem Level des Separators abhängig. Für einen vertikalen Separator vom Level l ist die Distanz $R_l = \frac{4 \cdot 4^l + 2}{3}$ und für einen horizontalen Separator $U_l = \frac{2 \cdot 4^l + 1}{3}$ (vgl. Lemma 2.1, S. 10). An Separatoren vom Level l ergibt sich eine mittlere logarithmische Distanz von

$$\mathcal{D}_{\text{avg}}^{\text{Lebesgue}}(l) = \frac{1}{2} \left(\log \left(U_l + 1 \right) + \log \left(R_l + 1 \right) \right) \ .$$

In Kombination mit der Wahrscheinlichkeit $p(l)$ für den Level eines Separators gilt

$$\mathcal{D}_{\text{avg}}^{\text{Lebesgue}} = \sum_{l=0}^{\infty} \mathcal{D}_{\text{avg}}^{\text{Lebesgue}}(l) \cdot p(l) \tag{4.14}$$

für unendlich große Gitter.

Es existiert kein geschlossener Term für (4.14). Um geeignete Schranken für den Term zu erhalten, können die ersten k Einträge der Summe bestimmt werden. Zusätzlich ist ein Fehlerterm für die fehlenden Summanden zu berechnen. Für Level $l \geq 3$ gilt

$$\begin{aligned} \mathcal{D}_{\text{avg}}^{\text{Lebesgue}}(l) &= \frac{1}{2} \left[\log \left(\frac{2 \cdot 4^l + 1}{3} + 1 \right) + \log \left(\frac{4 \cdot 4^l + 2}{3} + 1 \right) \right] \\ &< \frac{1}{2} \log \left(4^{2l} \right) = 2l \ . \end{aligned}$$

Für die Summe über die fehlenden Summanden erhält man

$$\sum_{l=k+1}^{\infty} \mathcal{D}_{\text{avg}}^{\text{Lebesgue}}(l) \cdot \frac{1}{2^{l+1}} < \sum_{l=k+1}^{\infty} \frac{2l}{2^{l+1}} = 2\frac{k+2}{2^k} \ . \tag{4.15}$$

Die Auswertung der Summanden 0 bis 17 ergibt 2.74583.... Die Auswertung von (4.15) ergibt einen Fehler kleiner als $0.16 \cdot 10^{-3}$ nach dem 17. Summanden. □

4.5.3.2 Hilbert-Kurve

In der Hilbert-Kurve gelten nicht die gleichen Symmetrieeigenschaften wie in der Lebesgue-Kurve, da während der Verfeinerung Rotationen und Spiegelungen der Basisstruktur durchgeführt werden. Insbesondere ist die Index-Distanz nicht nur von dem Level eines Separators sondern vielmehr auch von der Zellen-Position an diesem Separator abhängig. Damit ist die oben genutzte Beweisidee in diesem Fall nicht anwendbar. Hier erfolgt der Beweis für eine obere Schranke über die Selbstähnlichkeit der Kurve.

Satz 4.2 *Für die mittlere logarithmische Index-Distanz in der Hilbert-Kurve gilt*

$$\mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 3.25 \ .$$

Beweis: Aufgrund der Selbstähnlichkeit der Hilbert-Kurve ist es möglich, in einer gröberen Struktur Schranken der Index-Distanzen für die feinere originale Kurve zu finden. Der Beweis basiert auf der Struktur die in Abbildung 4.14 (links) dargestellt ist und folgt dem Ansatz in [28] zur Abschätzung des euklidischen Abstands in der Hilbert-Kurve. Innerhalb eines quadratischen Bereiches der Größe $4 \cdot 4^l$ ergibt sich immer die

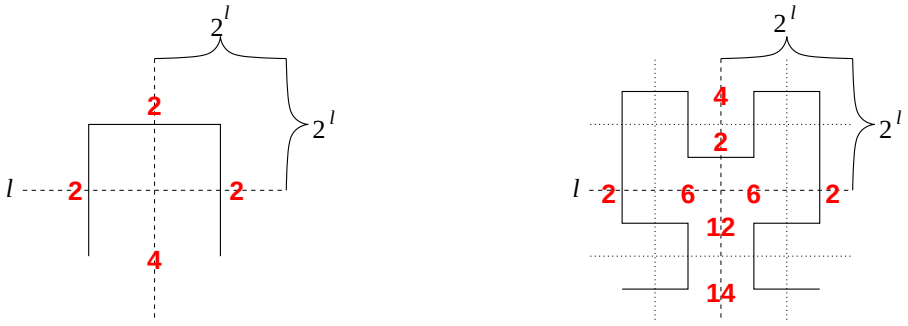


Abbildung 4.14: Größte Index-Distanzen in einer Hilbert-Kurve am Separator vom Level l in Blöcken der Größe $2^l \times 2^l$ (links) bzw. $2^{l-1} \times 2^{l-1}$ (rechts)

gleiche 'U'-Struktur mit den mittleren Separatoren vom Level l. Für die Zellen, die an den vier Teilseparatoren (Grenzen zwischen zwei Vierteln) angrenzen, kann die Index-Distanz abgeschätzt werden. An drei Teilseparatoren ist die Index-Distanz durch $2 \cdot 4^l$ beschränkt. Am vierten Teilseparator ist die Index-Distanz lediglich durch die Anzahl

aller Zellen, $4 \cdot 4^l$ beschränkt. Zusammen mit der Wahrscheinlichkeit für das Auftreten eines Separators vom Level l erhält man

$$\begin{aligned} \mathcal{D}_{\text{avg}}^{\text{Hilbert}} &< \sum_{l=0}^{\infty} \left(\frac{1}{4} \left[3 \cdot \log(2 \cdot 4^l) + 1 \cdot \log(4 \cdot 4^l) \right] \cdot \frac{1}{2^{l+1}} \right) \\ &= \frac{1}{4}(3 \cdot 3 + 4) = 3\frac{1}{4} \ . \end{aligned}$$

□

Der obige Beweisansatz kann auf feinere Gitter erweitert werden. Die zu untersuchende Struktur des nächstfeineren Gitters ist in Abbildung 4.14 (rechts) angegeben. Der quadratische Bereich der Größe $4 \cdot 4^l$ Zellen ist in 4×4 Blöcke aufgeteilt. Eine vergleichbare Auswertung vernachlässigt alle Index-Distanzen an Separatoren des Levels 0, die gesondert betrachtet werden müssen.

Aus den Distanzen in Abbildung 4.14 (links) ergibt sich eine mittlere Index-Distanz an Separatoren vom Level 0 von

$$\mathcal{D}_{\text{avg}}^{\text{Hilbert}}(0) = \frac{5}{4} \ .$$

Damit kann die mittlere Index-Distanz aller Separatoren bestimmt werden als

$$\begin{aligned} \mathcal{D}_{\text{avg}}^{\text{Hilbert}} &< \sum_{l=1}^{\infty} \left(\frac{1}{8} \left[3 \cdot \log(2 \cdot 4^{l-1}) + 1 \cdot \log(4 \cdot 4^{l-1}) + 2 \cdot \log(6 \cdot 4^{l-1}) \right. \right. \\ &\qquad \left. \left. +1 \cdot \log(12 \cdot 4^{l-1}) + 1 \cdot \log(14 \cdot 4^{l-1}) \right] \cdot \frac{1}{2^{l+1}} \right) + \frac{1}{2} \mathcal{D}_{\text{avg}}^{\text{Hilbert}}(0) \\ &= 2.0976 \ldots + \frac{5}{8} < 2.7226 \ . \end{aligned}$$

Mit Hilfe der gleichen Strategie kann die obere Schranke weiter verbessert werden:

$$\begin{aligned} 2 \times 2 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 3.25 \\ 4 \times 4 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 2.723 \\ 8 \times 8 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 2.580 \\ 16 \times 16 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 2.542 \end{aligned}$$

Eine computergestützte Berechnung der mittleren logarithmischen Index-Distanz ergibt für das 4096×4096 Gitter einen Wert von $2.524 \ldots$ (vgl. Abb. 4.13). Zusammen mit dem Ergebnis aus Lemma 4.1 und den oben aufgeführten schärferen oberen Schranken folgt:

Satz 4.3 *Die mittlere logarithmische Index-Distanz der Hilbert-Kurve liegt in dem Bereich*

$$2.524 < \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < 2.542 \ .$$

4.5.3.3 $\beta\Omega$-Indizierung

Für die $\beta\Omega$-Indizierung gelten ähnliche Bedingungen wie für die Hilbert-Kurve. Durch Rotationen und Spiegelungen während der rekursiven Konstruktion der Kurve ist es nicht möglich, die mittlere logarithmische Index-Distanz exakt anzugeben. Der Ansatz zur Analyse der Index-Distanz ist jedoch der gleiche. Da die Grundstrukturen der Kurven für eine einzelne rekursive Verfeinerung identisch sind (U-Form), ergibt sich für die Analyse von 2×2 Blöcken das gleiche Resultat von

$$\mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 3.25 \ .$$

Bei der Betrachtung feingranularer Kurven muss zusätzlich beachtet werden, dass es zwei verschiedene Ableitungsregeln gibt. Die entsprechenden Index-Distanzen im Gitter mit 4×4 Blöcken sind in Abbildung 4.15 dargestellt. Die mittlere Index-Distanz an Separatoren vom Level 0 sind für Hilbert-Kurve und $\beta\Omega$-Indizierung identisch, mit

$$\mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}}(0) = \mathcal{D}_{\text{avg}}^{\text{Hilbert}}(0) = \frac{5}{4} \ .$$

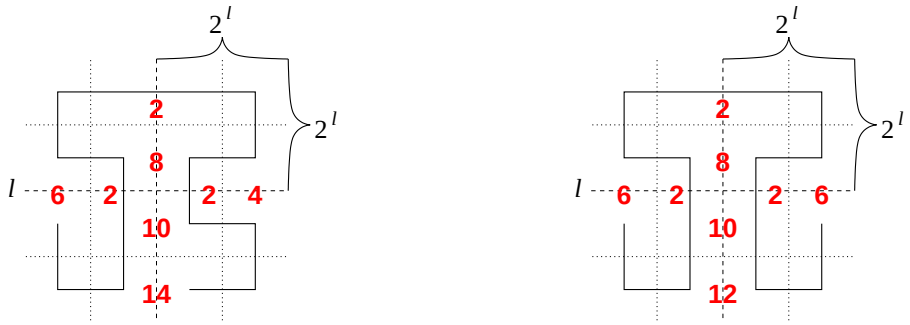


Abbildung 4.15: Größte Index-Distanzen am Separator vom Level l in Blöcken der Größe $2^{l-1} \times 2^{l-1}$ für beide Verfeinerungsregeln der $\beta\Omega$-Indizierung

Das Verhältnis in dem die beiden Substrukturen vorkommen konvergiert gegen $4:1$ in großen Gittern (vgl. Lemma 2.3, S. 18). Damit kann die mittlere Index-Distanz aller Separatoren bestimmt werden als

$$\begin{aligned}
\mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} \quad < \quad & \frac{4}{5}\sum_{l=1}^{\infty}\left(\frac{1}{8}\left[3\cdot\log(2\cdot 4^{l-1}) + \log(4\cdot 4^{l-1}) + \log(6\cdot 4^{l-1})\right.\right. \\
& \qquad \left.\left. + \log(8\cdot 4^{l-1}) + \log(10\cdot 4^{l-1}) + \log(12\cdot 4^{l-1})\right]\frac{1}{2^{l+1}}\right) \qquad (4.16) \\
& + \frac{1}{5}\sum_{l=1}^{\infty}\left(\frac{1}{8}\left[3\cdot\log(2\cdot 4^{l-1}) + 2\cdot\log(6\cdot 4^{l-1}) + \log(8\cdot 4^{l-1})\right.\right. \\
& \qquad \left.\left. + \log(10\cdot 4^{l-1}) + \log(12\cdot 4^{l-1})\right]\frac{1}{2^{l+1}}\right) \qquad (4.17) \\
& + \frac{1}{2}\mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}}(0) \\
= \quad & 2.112\ldots + \frac{5}{8} < 2.737 \ . \qquad (4.18)
\end{aligned}$$

Dabei bezieht sich der Term (4.16) auf die *gebogene* und der Term (4.17) auf die *gerade* Substruktur.

Wiederum kann diese Analyse für feinere Strukturen durchgeführt werden. Je nach Granularität der gewählten Struktur ergeben sich folgende obere Schranken:

$$\begin{aligned} 2 \times 2 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 3.25 \\ 4 \times 4 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 2.723 \\ 8 \times 8 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 2.580 \\ 16 \times 16 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 2.542 \\ 32 \times 32 \text{ Blöcke} &\Rightarrow \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 2.513 \end{aligned}$$

Satz 4.4 *Die mittlere logarithmische Index-Distanz der $\beta\Omega$-Indizierung liegt in dem Bereich*

$$2.503 < \mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < 2.513 \ .$$

4.5.3.4 Zusammenfassung

Die Ergebnisse zur mittleren logarithmischen Index-Distanz sind in Tabelle 4.7 zusammengefasst [82]. Für die H-Indizierung sind bisher keine Schranken bekannt. Um die Qualität dieser Kurve dennoch einschätzen zu können, ist ein Wert angegeben, der sich aus den experimentellen Ergebnissen in Abbildung 4.13 ableiten lässt. Zudem muss beachtet werden, dass das Maß in diesem Fall keine so große Aussagekraft für die Qualität in der globalen Kontaktsuche hat, da es sich nicht um eine rekursive raumfüllende Kurve handelt.

Indizierungs-schema	mittlerer Fall	
	untere S.	obere S.
Hilbert	2.524	2.542
Lebesgue	2.745	2.746
H-Indizierung	≈ 2.58	
$\beta\Omega$-Indizierung	2.503	2.513

Tabelle 4.7: Logarithmische Index-Distanz im mittleren Fall

Ähnliches gilt für eine zeilenweise Nummerierung. Die Komplexität der logarithmischen Index-Distanz liegt in $O(\log n)$, da die Hälfte der benachbarten Zellen eine Index-Distanz von n haben. Die Analyse in Abschnitt 4.4 zeigt jedoch, dass die Algorithmen in diesem Fall eine gänzlich andere Komplexität aufweisen, die nicht nur mit der Index-Distanz, sondern auch mit anderen Eigenschaften dieser Codierung zusammenhängen.

Für die untersuchten rekursiven raumfüllenden Kurven lässt sich aus den bewiesenen Schranken eine Ordnung bzgl. der logarithmischen Index-Distanz angeben:

Korollar 4.5 *Für die mittlere logarithmische Index-Distanz in raumfüllenden Kurven gilt die Ordnung*

$$\mathcal{D}_{\text{avg}}^{\beta\Omega-\text{Ind}} < \mathcal{D}_{\text{avg}}^{\text{Hilbert}} < \mathcal{D}_{\text{avg}}^{\text{Lebesgue}} \quad .$$

Beweis: Die Ordnung ergibt sich direkt aus den Sätzen 4.1, 4.3 und 4.4. □

Aus der Ordnung der Kurven gemäß ihrer logarithmischen Index-Distanz folgt, dass die Hilbert- und die $\beta\Omega$-Kurve für die globale Kontaktsuche besser geeignet sind als die Lebesgue-Kurve. Der Einfluss der Differenzen in diesen theoretischen Ergebnissen, die wiederum von einem abstrahierten Szenario ausgehen, wird jedoch in realen Anwendungen kaum messbar sein. Zudem ist die Struktur der Lebesgue-Kurve deutlich einfacher, so dass die Algorithmen überschaubarer implementiert werden können. Insbesondere das Aktualisieren der Codes nach einer Verschiebung von Knoten erfordert bei komplexeren raumfüllenden Kurven ein deutlich tieferes Verständnis für deren Konstruktion.

Kapitel 5

Lastverteilung und Kontaktbehandlung in einer industriellen Applikation

Die Ergebnisse der Kapitel 3 und 4 sind in Teilen in ein Projekt mit der Hilti AG eingeflossen. Zielsetzung des Projektes war zum einen die Parallelisierung einer vorhandenen Applikation zur expliziten FE-Simulation in der Strukturmechanik. Zum anderen sind algorithmische Erweiterungen implementiert worden, die den Einsatzbereich der Simulationsumgebung vergrößern. Um diese Ziele zu erreichen, sind u. a. die vorgestellten Verfahren zur Partitionierung und Kontaktsuche über die Struktur raumfüllender Kurven umgesetzt worden.

Für Applikationen im industriellen Umfeld stehen neben der kurzen Laufzeit für die Simulation einer gegebenen Problemstellung auch die Stabilität und Korrektheit der Implementation effizienter Algorithmen im Vordergrund. Für die Problemanalyse und Fehlersuche innerhalb komplexer Anwendungen ist es hilfreich, einfachere Methoden umzusetzen, deren Umsetzung in übersichtlichen Strukturen möglich ist. Daher bieten sich die vorgestellten Verfahren auf Basis der raumfüllenden Kurven an.

Die Parallelisierung für Shared-Memory-Systeme mit der dynamischen Lastverteilung über die implizite Partitionierung führt zu guten Skalierungseigenschaften auch bei kleinen Eingaben. Die integrierte allgemeine Kontaktbehandlung erlaubt die Simulation von adaptiven Netzoberflächen, die bei der Rissbildung (Fragmentierung) oder der adaptiven Neuvernetzung auftreten. Aufgrund der hohen Effizienz der entwickelten Methode zur globalen Kontaktsuche, verringert die sequentielle Kontaktbehandlung die Skalierbarkeit der Anwendung nur leicht.

5.1 Eigenschaften

Die zugrunde liegende Applikation wird zur zweidimensionalen Simulation strukturmechanischer Prozesse in den Bereichen Abbauen, Befestigen und Schneiden einge-

setzt. Dabei ist insbesondere das Materialverhalten an den Oberflächen der simulierten Objekte von großem Interesse. Dadurch ergeben sich besondere Anforderungen an die Qualität der Kontaktbehandlung und die Netzdichte an den Gebietsrändern. Um die Wellenausbreitung im Material exakt nachbilden zu können, sind oftmals mehrere 100 000 Zeitschritte der expliziten Zeitintegration erforderlich. Die Kombination aus der Anzahl an Elementen und Zeitschritten führt zu einer sequentiellen Rechenzeit von einigen Tagen bei relevanten Eingabeinstanzen. Ziel des Projektes war die Reduktion der Rechenzeit auf vier bis acht Stunden.

Neben der Berechnung der inneren Materialeigenschaften beinhaltet die Applikation auch Routinen zur adaptiven Neuvernetzung, zur Kontaktbehandlung und zur Berücksichtigung von vorgegebenen Randbedingungen. Die adaptive Neuvernetzung erlaubt die Neugenerierung des Netzes eines simulierten Körpers, falls die geometrische Qualität die numerische Stabilität einschränkt. Anhand der Spannungs- und Dehnungszustände wird ein neues Netz generiert, dessen Dichte an den Gradienten der Integrationspunktinformationen ausgerichtet ist.

Für die Integration der Fragmentierung im Material, die im Abbauprozess von großer Bedeutung ist, ist eine Kontaktbehandlung erforderlich, die alle Kontakte physikalisch korrekt löst. Die notwendige robuste Kontaktkorrektur wird durch eine vollständige geometrische Analyse aller Kontaktsituationen innerhalb der lokalen Kontaktsuche erreicht. Um die guten Skalierungseigenschaften der parallelen Applikation zu erhalten, ist zudem eine effiziente globale Kontaktsuche unumgänglich.

Die Applikation erlaubt die Berücksichtigung einer Vielzahl von Eigenschaften, die auf die Knoten oder Kanten an der Oberfläche der diskretisierten Körper einwirken. Diese werden als Randbedingungen bezeichnet. Neben der Lagerung der Knoten in achsenorientierter Richtung (eine oder beide) ist es möglich, variable Geschwindigkeiten an Knoten vorzugeben. Zudem kann an Randgebieten ein variabler Druck definiert werden.

5.2 Dynamische Lastverteilung

Für eine dynamische Lastverteilung ist die Bestimmung der Elemente notwendig, die zwischen den Partitionen ausgetauscht werden. Für die Entscheidung, welche Elemente auszutauschen sind, werden Heuristiken eingesetzt, deren Ziel es ist, den Kantenschnitt gering zu halten. Bei diesen Heuristiken existiert ein Tradeoff zwischen der erreichbaren Qualität und der Laufzeit zur Erzeugung der Partitionierung. Sind sehr strenge Anforderungen an die Laufzeit gestellt, da wie im vorgestellten Fall eine häufige Lastverteilung notwendig ist, wird die Qualität der Partitionierung im Verlauf der Berechnung deutlich abnehmen. Bei der impliziten Partitionierung ergeben sich die auszutauschenden Elemente direkt, sobald die Menge der auszutauschenden Last bestimmt ist. Zudem bleibt die erwartete Qualität im gesamten Verlauf der Berechnung konstant.

Die Art der Modellierung der Materialeigenschaften und die daraus resultierenden Rechenaufwände machen diese Applikation zu einem geeigneten Anwendungsgebiet für

eine implizite Partitionierung. Insbesondere die spröden Materialien, die teilweise auch die Simulation von Fragmentierung erlauben, sind nichtlinear modelliert. Damit ergeben sich schwankende Rechenlasten je Element, die vom momentanen Zustand des Elementes abhängen. Diese Schwankungen können oszillierend oder auch in Bereichen monoton verlaufen. Sie sind in jedem Fall nicht vorhersagbar und fallen teilweise sehr stark aus. Daher ist der Einsatz einer statischen Lastverteilung in dieser Art von Applikation nicht sinnvoll.

5.2.1 Parallelisierung

Abbildung 5.1 skizziert die Datenabhängigkeiten auf den grundlegenden Datenstrukturen. Die zu simulierenden Körper sind über ein FE-Netz aus Knoten und Elementen diskretisiert. Daten, die die aktuelle Materialeigenschaften abbilden (z. B. Dehnungen und Spannungen) werden an Integrationspunkten gespeichert, die an festen relativen Positionen im Innern der Elemente liegen. Die geometrischen Eigenschaften (z. B. Verschiebung oder Geschwindigkeit) sind den Knoten zugeordnet.

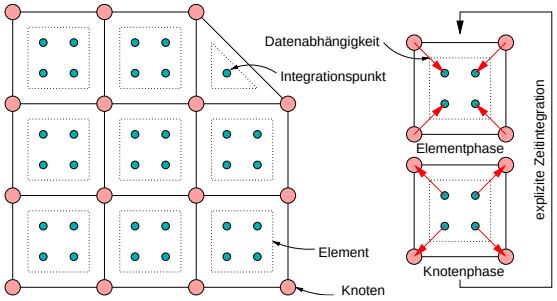


Abbildung 5.1: Struktur der expliziten FE-Applikation

Jeder Schritt der expliziten Zeitintegration teilt sich in die Phasen der Aktualisierung von Element- und Knotenzuständen. Bei der Elementaktualisierung werden die aktuellen Dehnungen anhand der anliegenden Knotenpositionen bestimmt. Aus den Dehnungen ergeben sich über die Materialgesetze die entsprechenden Spannungen im Element. Daraus resultieren wiederum innere Kräfte auf die angrenzenden Knoten. Die Berechnung der Knoteneigenschaften wird in der Phase der Knotenaktualisierung durchgeführt. Dazu werden zu den inneren Kräften die entsprechenden Beschleunigungen berechnet aus denen die Geschwindigkeit und Verschiebung mittels expliziter Zeitintegration resultieren.

Die gegebene Applikation ist über Threads parallelisiert, die über Daten im gemeinsamen Adressraum kommunizieren. Die Synchronisation der Bearbeitung erfolgt über Barrier. Wesentliche Gründe für die Wahl dieser Art der Parallelisierung bestehen in der einfacheren Umsetzung dieses Ansatzes und der Möglichkeit der schrittweisen Erweiterbarkeit. Sequentielle und parallele Phasen können nacheinander abgearbeitet werden, ohne das explizit die notwendigen Daten eingesammelt und im nachhinein wieder verteilt werden müssen. Somit können algorithmisch komplexe aber wenig rechenintensive

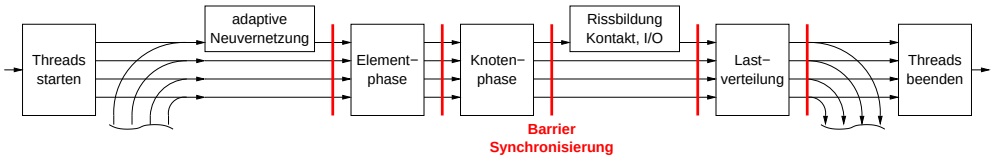


Abbildung 5.2: Programmfluss der parallelen Version

Phasen sequentiell belassen werden. Neue Algorithmen, die den Funktionsumfang der Applikation erweitern, können zunächst in einer sequentiellen Form implementiert werden. Erst wenn das Verhalten des Algorithmus verifiziert ist und sich zeigt, dass die Skalierbarkeit der Gesamtapplikation durch diesen deutlich eingeschränkt wird, ist eine anschließende Parallelisierung sinnvoll.

An dieser Stelle werden lediglich die für die Interpretation der Ergebnisse notwendigen Details beschrieben. Weitergehende Informationen zur Parallelisierung der Applikation finden sich in [46]. Der Programmverlauf der parallelen Applikation ist in Abbildung 5.2 angegeben. Die parallelen Linien geben den Kontrollfluss der nebenläufigen Threads wieder. Die Threads werden zu Beginn der Berechnung gestartet und bleiben während der gesamten Rechnung aktiv. Einige Phasen werden von allen, andere nur von einem einzelnen Thread bearbeitet.

Die berechnungsintensiven Phasen von Element- und Knotenaktualisierung werden parallel verarbeitet. Beim Aktualisieren der Integrationspunkte der Elemente sind die schreibenden Zugriffe lokal und lediglich lesende Zugriffe auf knotenassoziierte Daten notwendig. Daher bietet sich in dieser Phase eine Partitionierung der Elemente an. In der anderen Phase sind die Schreibzugriffe knotenlokal, so dass eine Partitionierung der Knoten erforderlich ist. Zwischen diesen beiden Phasen ist eine Barrier-Synchronisation notwendig, da die Lesezugriffe auf elementbezogenen Daten während der Knotenaktualisierung erst durchgeführt werden, wenn sichergestellt ist, dass die Schreiboperationen auf diesen abgeschlossen sind. Für Elemente und Knoten ergibt sich eine i. A. unabhängige Partitionierung. Für beide wird initial eine Ordnung bzgl. der Hilbert-Kurve bestimmt, die für die implizite Partitionierung eingesetzt wird. Da der Berechnungsaufwand in der Phase zur Aktualisierung der Knotenzustände konstant ist, bleibt die Partitionierung fest, solange keine neuen Knoten durch die adaptive Netzanpassung auftreten. Damit ist die dynamische Lastbalancierung auf die Elemente des Netzes beschränkt.

Die Lastbalancierung zwischen den Partitionen erfolgt gemäß der jeweiligen Rechenlast, die durch eine Zeitmessung bestimmt wird. Dabei werden für jedes Element innerhalb einer Partition die gleichen Berechnungskosten angenommen.[1] In Abbildung 5.3 ist der Ablauf der Lastbalancierung an einem Beispiel skizziert. Die *aktuelle Last* der einzelnen Elemente ist in Form eines Histogrammes gegeben und liegt im Bereich von eins bis fünf. Abweichend von dieser *aktuellen Last*, wird nur eine *gemittelte Last* je Element

[1] Experimentelle Untersuchungen haben gezeigt, dass die Zeitmessung für einzelne Elemente einen zu großen Overhead verursacht. Zudem ist es nicht sinnvoll, bei einer so stark schwankenden Lastsituation, mit derart feingranularen Werten zu arbeiten.

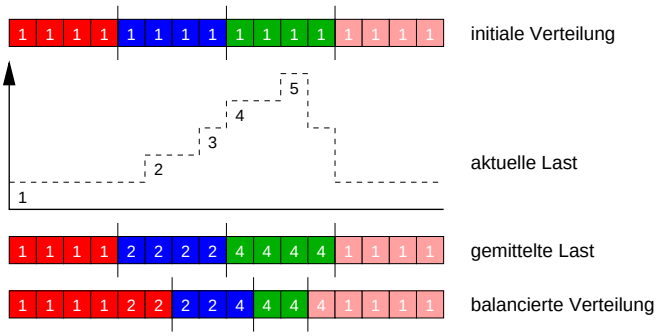


Abbildung 5.3: Anpassung der Lastintervalle bei der Lastbalancierung

für jede Partition bestimmt. Auf Basis dieser Näherung wird durch die Verschiebung der Indexgrenzen innerhalb der geordneten Elementfolge eine neue Partitionierung festgelegt. In dem gegebenen Beispiel wird eine gleichmäßige Last von acht Einheiten je Partition erreicht. Unter Berücksichtigung der tatsächlichen Lastsituation haben die Partitionen jedoch Lasten zwischen sieben und neun. Es hat sich jedoch gezeigt, dass die Lastdifferenzen innerhalb der Applikation nicht signifikant sind und zudem in den folgenden Iterationen weiter ausgeglichen werden.

Die Kombination aus einer Parallelisierung basierend auf Threads und der schnellen dynamischen Lastverteilung mittels impliziter Partitionierung erlaubt die Erweiterung des parallelen FE-Codes zu einer *malleablen* Applikation. Während der Laufzeit kann der Grad der Parallelität verändert werden, indem weitere Threads gestartet oder aber terminiert werden. In diesem Fall ist die Steuerung der Anzahl der Threads entweder über Prozess-Signale oder einen zentralen Scheduler möglich, der die vorhandenen Ressourcen gemäß verschiedener Strategien verteilt, um das System effizient zu nutzen [40].

5.2.2 Ergebnisse

Die Art der Parallelisierung in Kombination mit der flexiblen Partitionierung basierend auf der Hilbert-Kurve führt zu einer hohen Effizienz bis zu einer moderaten Anzahl von Prozessoren. Die Applikation wird auf zwei HP 9000/800 Systemen mit jeweils 16 CPUs vom Typ PA 8600 mit 450 MHz eingesetzt. Auf Vorgängermodellen vom Typ HP X-Class skalierte eine ältere Version der Applikation auf bis zu 48 parallelen Threads mit hoher Effizienz. Die folgende Analyse beschränkt sich jedoch auf Messwerte die auf den aktuell eingesetzten Systemen erzielt werden.

Abbildung 5.4 zeigt die für diese Analyse ausgewählten Benchmarks. Die Benchmarks tragen im Folgenden die Bezeichnungen *Meißel* (oben links), *Zylinder* (oben rechts) und *Bolzen* (unten). Bei den Simulationen Meißel und Zylinder wird ein Werkzeug aus Stahl mit einer vorgegebenen Anfangsgeschwindigkeit auf einen Untergrund aus Beton bewegt. Die Modellierung des Betons erlaubt Fragmentierung, so dass im Verlauf der Berechnung neue Oberflächen entstehen. Der rechenintensive Anteil liegt im Bereich

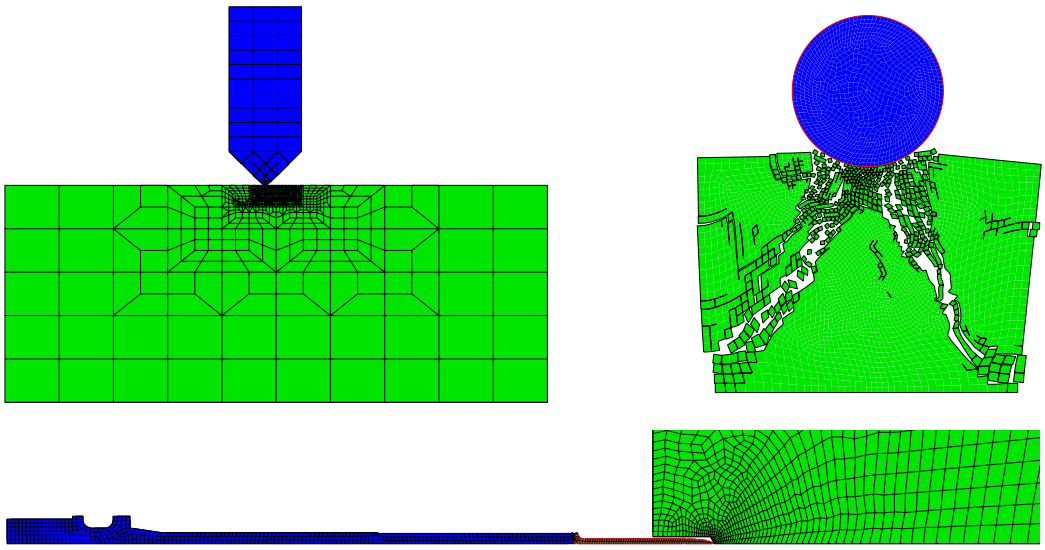


Abbildung 5.4: Benchmarks: Meißel, Zylinder und Bolzen

des Betons, der durch ein nichtlineares Materialgesetz beschrieben ist. Der Benchmark Bolzen besteht aus drei Geometrien: einem Kolben, der mit einer vorgegebenen Anfangsgeschwindigkeit den Bolzen in den Untergrund eintreibt. Kolben und Bolzen sind elastisch, der Untergrund thermoplastisch modelliert. Zudem wird der Untergrund adaptiv neu vernetzt, wenn die Verformungen der finiten Elemente die numerische Genauigkeit beeinträchtigen.

In Tabelle 5.1 sind die wesentlichen charakteristischen Eigenschaften der Benchmarks angegeben. Aufgrund der Neuvernetzung im Beispiel Bolzen schwankt die Netzgröße im Verlauf der Rechnung. Die Intervalle geben jeweils die minimale und maximale Größe an.

Netz	Knoten	Elemente	Integrationspunkte
Meißel	1 136	1 099	4 291
Zylinder	3 981	3 839	15 356
Bolzen	4 719 – 5 376	4 448 – 5 098	17 792 – 19 812

Tabelle 5.1: Eigenschaften der Benchmarks

Abbildung 5.5 zeigt den Verlauf der Partitionsgröße für einen Teil der Simulation des Benchmarks Meißel. Dabei sind die Integrationspunkte der Partitionen akkumuliert dargestellt, so dass der Abstand zwischen zwei Kurven die Größe der jeweiligen Partition angibt. Die Veränderungen der Partitionsgröße beinhalten Schwankungen unterschiedlicher Breite, die sich gegenseitig überlagern. Zum einen gibt es Schwankungen innerhalb einzelner Zeitschritte, zum anderen welche über mehrere tausend Zeitschritte. In jedem Fall ist keine einheitliche Tendenz zu erkennen, welche die Möglichkeit einer Vorhersage der Rechenlast über mehrere Zeitschritte erahnen lässt. Diese Ergebnisse zeigen, dass durch die nichtlineare Modellierung von Materialien sich schnell und nicht

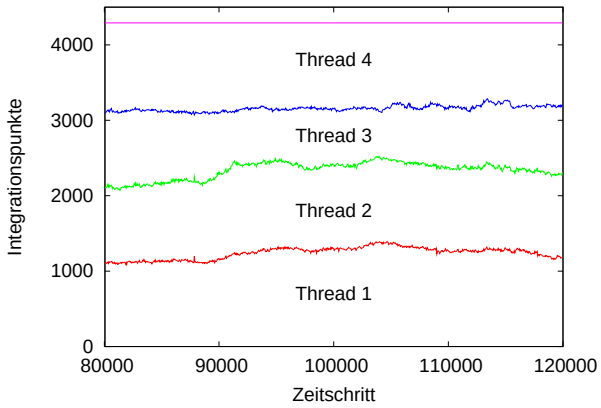


Abbildung 5.5: Variierende Partitionsgröße beim Benchmark *Meißel*

vorhersehbar ändernde Rechenlasten ergeben. Diese erfordern eine häufige Anpassung der Partitionierung auf der Basis schneller Algorithmen.

Innerhalb des untersuchten Intervalls reduziert die dynamische Lastverteilung die Partition von Thread 3 auf etwa die Hälfte. Die Elementphase erfordert für diese Elemente einen hohen Aufwand. Ohne die regelmäßige Lastverteilung ergibt sich ein erhöhter Rechenaufwand, der die gesamte parallele Applikation verlangsamt. Dieser Effekt ist in Abbildung 5.6 dargestellt. Bis zum Zeitschritt 80 000 ist die adaptive Lastverteilung aktiv, danach ist sie zur Demonstration deaktiviert, so dass die Partitionsgrößen für den weiteren Verlauf der Rechnung konstant bleiben. Die Laufzeiten der vier nebenläufigen Threads sind farbig dargestellt.

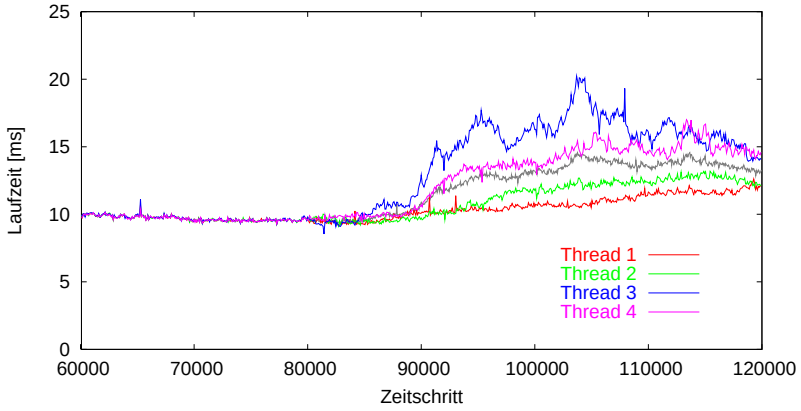


Abbildung 5.6: Laufzeiten der Threads bei statischer Lastverteilung ab Schritt 80 000

Während der dynamischen Lastverteilung bis zum Zeitschritt 80 000, haben alle vier Threads nahezu identische Laufzeiten für jeden einzelnen Schritt der Zeitintegration. Ab dem 90 000sten Zeitschritt erhöht sich die Last von Thread 3 deutlich. Aufgrund der Synchronisation zwischen den Threads erhöht sich damit auch die Laufzeit der gesamten Applikation. Die Laufzeit der Threads bei weiterhin dynamischer Lastverteilung ist in grau dargestellt. Sie liegt im gezeigten Intervall immer unter 15 Millisekunden. Damit

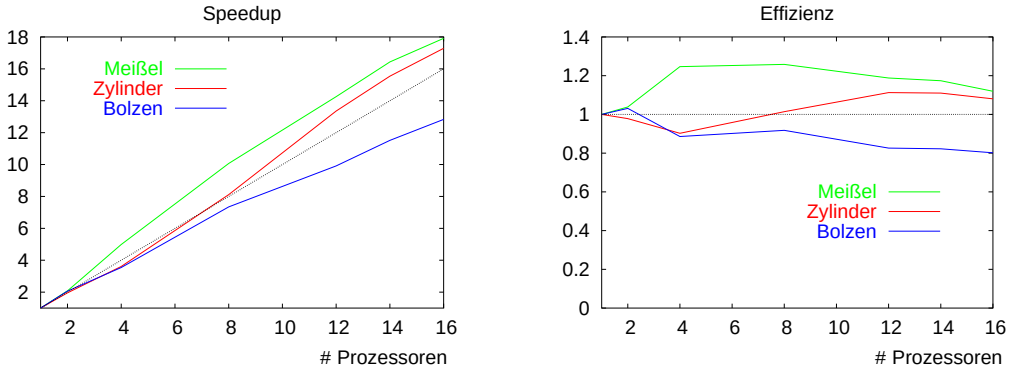


Abbildung 5.7: Speedup (links) und Effizienz (rechts) der parallelen Applikation

liegt die Rechenzeit für einen Schritt der Zeitintegration um bis zu 40 Prozent höher, wenn keine regelmäßige Neuverteilung der Rechenlast durchgeführt wird.

Die Gradienten der Rechenzeit zeigen, dass die Lastverteilung nach einigen hundert Zeitschritten zwingend notwendig ist. Bei dem gegebenen Beispiel folgt daraus ein Lastausgleich bereits nach wenigen Sekunden. Die beschriebene Problematik wächst mit der Anzahl der Rechenknoten: Die Auswirkungen der schwankenden Rechenlasten vergrößert sich, während die Rechenzeiten kürzer werden.

Abbildung 5.7 zeigt die Skalierbarkeit der parallelen Applikation. Die drei gewählten Benchmarks, welche unterschiedliche Charakteristika bzgl. Netzgröße und modelliertem Material aufweisen, skalieren bis zu 16 Prozessoren. Der Benchmark *Bolzen* erreicht einen Speedup von 13 und damit eine Effizienz von 80 Prozent. Die Berechnung des elastischen und plastischen Materials ist weniger aufwendig, wodurch sich ein hohes Verhältnis von Kommunikationsaufkommen zu Rechenaufwand ergibt. Die Qualität der Partitionierung ist jedoch ausreichend, um diese Effizienz zu erreichen. Die Simulation der spröden Materialien in den Benchmarks *Meißel* und *Zylinder* mit ihrer nicht-linearen Modellierung erlauben Speedups von 17.3 bzw. 17.9. Insbesondere durch die Möglichkeit einer schnellen regelmäßigen Lastbalancierung, die eine ständige Gleichverteilung der Last erlaubt, ergibt sich diese hohe Effizienz (108 % bzw. 112 %) bei verhältnismäßig kleinen Netzen.

Der superlineare Speedup ergibt sich aus Vorteilen bei der Nutzung der Daten-Caches. Die Nutzung mehrerer Prozessoren erhöht den der Applikation zur Verfügung stehenden schnellen Cache-Speicher. Mit wachsender Anzahl an Prozessoren kann ein immer größerer Anteil der notwendigen Daten im Cache gehalten werden, während nur die von mehreren Prozessoren gemeinsam genutzten Daten regelmäßig im zentralen Hauptspeicher abgelegt werden (Kommunikation). Der Speicherbedarf des Benchmarks Zylinder ist in etwa viermal so groß wie der des Benchmarks Meißel. Letzterer profitiert schon ab einer Anzahl von vier Threads vom größeren akkumulierten Cache-Speicher, während sich beim anderen Benchmark erst bei einer größeren Prozessoranzahl ein superlinearer Speedup ergibt.

5.3 Kontaktbehandlung

Dieser Abschnitt beschreibt die Kontaktbehandlung innerhalb der industriellen Applikation. Im Mittelpunkt steht dabei die Effizienz der vorgestellten Variante des Position-Code-Algorithmus für die Phase der globalen Kontaktsuche. Zudem werden die physikalischen Grundlagen der Kontaktkorrektur und dabei auch die Modellierung geglätteter Oberflächen vorgestellt. Ein weiterer wesentlicher Bestandteil ist die lokale Suche, in der die Entscheidung über die Korrekturrichtung getroffen wird. Die Korrektheit dieser Phase ist für die Robustheit der gesamten Kontaktbehandlung von großer Bedeutung. Die wesentlichen Ideen für eine systematische Erfassung aller Kontaktsituationen werden hier ebenfalls skizziert.

Teile der globalen Kontaktsuche sind im Rahmen einer Diplomarbeit in die Applikation integriert worden [50]. An dieser Stelle werden nur einige Details der Umsetzung in der Applikation gegeben, die für die Erläuterung der Benchmark-Ergebnisse von Bedeutung sind. Genaue Beschreibungen der Datenstrukturen und Routinen sind in vertraulichen technischen Berichten zu finden.

5.3.1 Kontaktformulierung

Die hier untersuchte FE-Applikation dient der Simulation zweidimensionaler strukturmechanischer Prozesse. Die Zeitintegration erfolgt über die Berechnung innerer Kräfte die auf Knoten wirken.[2] In dieser Phase werden mögliche Durchdringungen zugelassen, die in der nachfolgenden Phase der Kontaktkorrektur gelöst werden (vgl. Abschnitt 5.2.1). Diese besteht aus der Kontaktsuche, der Berechnung der Kontaktkräfte auf die betroffenen Knoten und der Korrektur der Zeitintegration bzgl. der Kontaktknoten [19].

Die Kontaktformulierung basiert auf den Beschreibungen in [5, 14, 75]. Der Algorithmus arbeitet innerhalb eines Zeitschritts rein geometrisch und ist damit für beliebige Materialien einsetzbar. Es ergibt sich ein sehr harter Kontakt, d. h. große Kräfte bei initialem Kontakt. Die Zeitintegration und die Zeitschrittweite müssen so gewählt werden, dass sie Oszillationen der Spannungen vermeiden und die Wellenausbreitung korrekt abgebildet wird. Die Anforderungen an den Kontaktalgorithmus beinhalten die Behandlung

- beliebiger Geometrien, auch mit Löchern,
- von Rissbildung mit neu auftretenden Oberflächen und scharfen Ecken,
- aller Kombinationen von eckigen und geglätteten Oberflächen,
- der Selbstdurchdringung von Oberflächen und
- der Lagerung von Oberflächenknoten.

[2] Innerhalb der Simulation wird angenommen, dass die Massen der Elemente anteilig an ihren Knoten konzentriert sind.

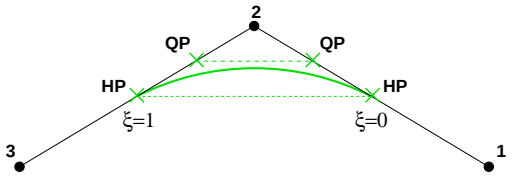


Abbildung 5.8: Beschreibung geglätteter Oberflächen

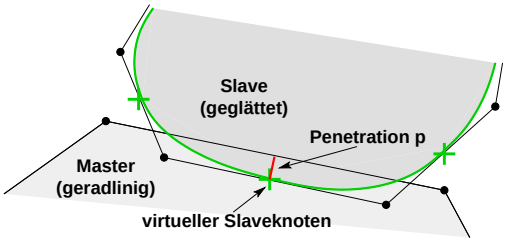


Abbildung 5.9: Penetration eines „virtuellen“ Slaveknotens in ein geradliniges Mastersegment

Die Definition eckiger (stückweise linearer) und geglätteter Oberflächen sind so gewählt, dass sie die obigen Anforderungen erfüllen. Die Geometrien sind über polynomielle Form-Funktionen h_i und Knotenvektoren $\mathbf{x}_i$ definiert:

$$\mathbf{x} = h_i(\xi) \cdot \mathbf{x}_i, \quad 0 \leq \xi < 1 \tag{5.1}$$

Die linearen Form-Funktionen $h_1^L = 1 - \xi$ und $h_2^L = \xi$ werden für die Beschreibung stückweise linearer Oberflächen mit Ecken eingesetzt. Die Formulierung für geglättete Oberflächen folgt den Vorschlägen von Wriggers, Taylor und Krstulović [76, 83]. Ein Oberflächensegment ist über drei adjazente Knoten des FE-Netzes durch Bézier-Splines definiert (vgl. Abb. 5.8):

$$h_1^B = \frac{1}{4}(2 - 3\xi + \xi^3), \qquad h_2^B = \frac{1}{4}(2 + 3\xi - 3\xi^2), \qquad h_3^B = \frac{1}{4}(3 - \xi)\xi^2 \tag{5.2}$$

Ein geglättetes Oberflächensegment beginnt an dem Mittelpunkt der ersten Netzkante ($\xi = 0$) und endet am Mittelpunkt der zweiten ($\xi = 1$). Die Oberfläche ist c_1-stetig und an den Mittelpunkten der Netzkanten tangential zur Netzoberfläche. Die so beschriebende Oberfläche verläuft zwischen den Verbindungslinien zwischen den Mittelpunkten (HP) bzw. zwischen den Viertelpunkten (QP) der Netzkanten. Die Vernetzung muss so fein gewählt werden, dass die Differenz zwischen dem Netz und der geglätteten Oberfläche klein ist. Damit ist es möglich, die Glättung bei der Berechnung der inneren Kräfte zu vernachlässigen. Penetrationen werden nur an den Kantenmittelpunkten überprüft, an denen die Positionen der Netzkanten und der geglätteten Oberflächen übereinstimmen. Dabei wird die Lage der Punkte gegenüber möglichen Kontaktsegmenten getestet (Master-Slave-Konzept). Das Mastersegment kann sowohl eine eckige als auch eine geglättete Oberfläche sein. An stückweise geradlinigen Oberflächen liegen die Slaves an den Knoten des FE-Netzes. An geglätteten Bereichen der Oberfläche werden „virtuelle“ Slaveknoten an den Mittelpunkten der Netzkanten angenommen, wie in Abbildung 5.9 dargestellt.

Im Falle der Selbstdurchdringung oder der Penetration an scharfen Ecken sind spezielle Algorithmen zur Bestimmung des Penetrationsvektors $\mathbf{p}$ notwendig. In den anderen Fällen ergibt sich der Penetrationsvektor über den kürzesten Abstand des Slaveknotens zum Mastersegment. In manchen Situationen erfordert die vollständige Erfüllung der

Nichtpenetrationsbedingung, dass in einer weiteren Phase die Zuordnung von Master und Slave getauscht wird. Dazu werden nacheinander *alle* Oberflächensegmente als Master angesehen und die jeweiligen potentiellen Kontaktpartner als Slaves.

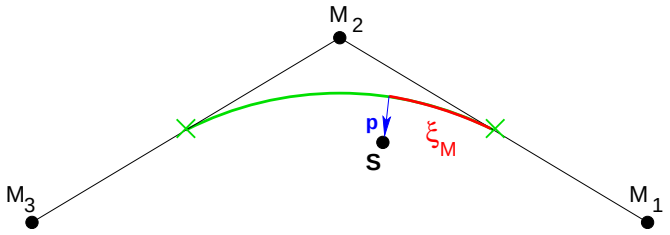


Abbildung 5.10: Bezeichner zur Kontaktkorrektur

Die Normalkraft wirkt in Richtung des Penetrationsvektors $\mathbf{p}$. Sei ξ_M die relative Position des Schnittpunktes von $\mathbf{p}$ mit dem Mastersegment und ξ_S die relative Position eines virtuellen Slaveknotens auf einer geglätteten Oberfläche (vgl. Abb. 5.10). Die linearisierte Nichtpenetrationsbedingung

$$||\mathbf{p}|| - \mathbf{u}_S \cdot \mathbf{e}_p + \mathbf{u}_M(\xi_M) \cdot \mathbf{e}_p = 0 \tag{5.3}$$

mit den Verschiebungen von Master ($\mathbf{u}_M$) und Slave ($\mathbf{u}_S$) kann eingesetzt werden, da bei der expliziten Zeitintegration i. A. sehr kleine Zeitschritte durchgeführt werden. Die Verschiebung auf Masterseite an der Penetrationsstelle ergibt sich aus

$$\mathbf{u}_M(\xi_M) \;=\; h_i(\xi_M) \cdot \mathbf{u}_{M_i} \;. \tag{5.4}$$

Die Knotenverschiebungen ergeben sich aus der Bewegungsgleichung

$$\mathbf{u}_S = \frac{1}{2}\frac{\Delta t^2}{m_S}\mathbf{F}_S, \qquad \mathbf{u}_{M_i} = \frac{1}{2}\frac{\Delta t^2}{m_{M_i}}\sum_j \mathbf{F}_{M_i} \;, \tag{5.5}$$

wobei j die Anzahl der beteiligten Slaveknoten ist. Im Bereich eckiger Oberflächen entspricht die Verschiebung der FE-Knoten denen der Slaveknoten. Im Falle geglätteter Oberflächen erfolgt die Berechnung der Knotenverschiebung analog zur Masteroberfläche:

$$\mathbf{u}_S = \mathbf{u}_S(\xi_S) = h_i(\xi_S) \cdot \mathbf{u}_{S_i} \;. \tag{5.6}$$

Die entsprechenden Gegenkräfte an den Masterknoten ergeben sich als

$$\mathbf{F}_{M_i} = h_i(\xi_M) \cdot \mathbf{F}_S \;. \tag{5.7}$$

Es ergibt sich ein lineares Gleichungssystem, dessen Größe von der Anzahl der aktiven Slaveknoten abhängt, die in benachbarte Mastersegmente eindringen. Diese Mastersegmente bilden gemeinsam mit den aktiven Slaveknoten eine *Kontaktgruppe*. Bei der Simulation zweier Objekte mit geglätteter Oberflächen ergibt sich möglicherweise nur eine Kontaktgruppe. Dagegen führt die Simulation diskreter Risse häufig zu einer großen Anzahl an kleinen Kontaktgruppen.

5.3.2 Datenstrukturen

Für die effiziente Simulation großer FE-Netze mit veränderlichen Oberflächen (Rissbildung, adaptives Vernetzen) ist die Integration einer flexiblen Datenstruktur unumgänglich. Die gemeinsame Simulation mehrere Objekte kombiniert mit der erforderten hohen Genauigkeit führt leicht zu Netzen mit mehreren Tausend Elementen und Knoten. Daher bilden alle Strukturen einen Kompromiss zwischen den Anforderungen an die Dynamik und dem schnellen direkten Zugriff auf die Daten. An dieser Stelle ist nur die Beschreibung der Oberflächen und Kontaktstrukturen erläutert, soweit es für die Beurteilung der Benchmark-Ergebnisse notwendig ist.

5.3.2.1 Oberflächen

Die zentrale Struktur zur Abbildung der Oberfläche des FE-Netzes ist eine doppelt verkettete Liste. Jeder Eintrag der Liste ist ein Repräsentant für einen Oberflächenknoten des Netzes. Die Ordnung der Verkettung ist so gewählt, dass bei einem Lauf über die Knoten der Liste in positiver Richtung das Material immer auf der linken Seite liegt. D.h. bei einer äußeren Oberfläche ergibt sich ein Umlauf in CCW-Orientierung *(counterclockwise)*. Jede einzelne Oberfläche eines Objektes ist zyklisch. Ein Eintrag der Oberflächenliste ist mit dem entsprechenden Eintrag der Knoten-Datenstruktur über Zeiger in beiden Richtungen gekoppelt.

Parallel zu der verketteten Liste existiert eine zweite Datenstruktur. Während erstere dem Verlauf des FE-Netzes folgt, repräsentiert die zweite die Oberflächenstruktur, über welche die Kontaktbehandlung erfolgt (*virtuelle Oberfläche*). Zwischen diesen Datenstrukturen findet die Unterscheidung von eckigen und geglätteten Bereichen der Oberfläche statt. Die virtuelle Oberfläche enthält Einträge für die charakteristischen Stellen der Oberfläche, über welche die Penetration getestet wird, d. h. die als Slave in einer Kontaktgruppe auftreten können. Für eckige Oberflächen stimmen diese mit den Knoten des Netzes überein, für geglättete Bereiche liegen sie auf den Mittelpunkten der Netzkanten, wie in Abbildung 5.11 dargestellt. Diese Stützstellen der Kontaktbehandlung sind mit den Einträgen der Oberflächenstruktur über Zeiger verbunden.

Ein Eintrag der virtuellen Oberfläche repräsentiert zwei Objekte: Einen Punkt auf der Oberfläche, der auf Penetration getestet wird *(Slaveknoten)* und ein Segment der Oberfläche *(Mastersegment)*, welches gerade oder gebogen sein kann. Die Form des Segmentes ergibt sich aus der Art des Eintrags und seine Orientierung aus der Verkettungsrichtung der Oberflächen-Datenstruktur.

5.3.2.2 Kontaktgruppen

Die potentiellen Kontaktpaarungen werden an den Mastersegmenten gespeichert. Jedes Oberflächensegment hat eine *Kandidatenliste* aus Slaveknoten, die in den folgenden k Schritten der Zeitintegration Kontakt haben können, und eine *Aktivenliste* der Slaveknoten, die im aktuellen Schritt eingedrungen sind. Mastersegmente werden zu

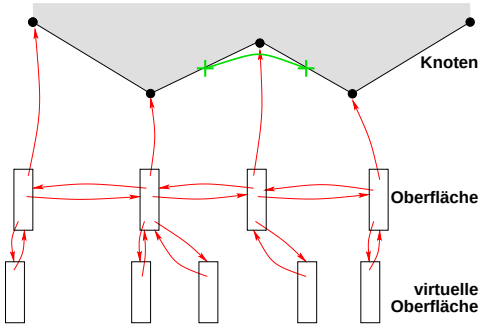


Abbildung 5.11: Verbindung des FE-Netzes mit der (virtuellen) Oberflächen-Datenstruktur (grün: geglättetes Oberflächensegment)

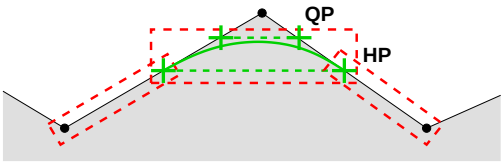


Abbildung 5.12: Form der Halos an geraden und geglätteten Oberflächensegmenten

Kontaktgruppen zusammengefasst, wenn über deren (potentiellen) Kontaktpartner eine Kopplung über die Korrekturgleichungen besteht (*Kandidatengruppe* bzw. *Aktivengruppe*). Die Kandidatenlisten werden im Laufe der globalen Suche aufgestellt und zu Gruppen zusammengefasst. Aus den Kandidaten werden die aktiven Slaveknoten in der lokalen Suche ausgewählt, und die Gruppen entsprechend verkleinert.

5.3.3 Globale Suche

In dieser Phase der Kontaktbehandlung werden die Kandidatenlisten für alle Oberflächensegmente und die zugehörigen Kandidatengruppen bestimmt. Dazu wird um jedes Segment ein Bereich (*Halo*) definiert, dessen enthaltene Knoten nah genug am Segment liegen, dass sie innerhalb weniger Zeitschritte eindringen können. Die Größe des Halos ist von der Art des Oberflächensegmentes abhängig. Im Falle eines geraden Segmentes wird eine ε-Umgebung um dieses gelegt. Im Falle eines gebogenen Segmentes wird eine ε-Umgebung um die Verbindungslinien der Mittelpunkte und der Viertelpunkte gelegt (vgl. Abb. 5.12). Der Wert von ε ist von der Diskretisierung des Netzes abhängig. Um eine Überlappung dreier benachbarter Halos zu vermeiden, ist ε kleiner als die Hälfte des kürzesten Oberflächensegmentes im gesamten FE-Netz. Ist $u_{\max}$ die größte Verschiebung eines Knotens je Zeitschritt, muss die folgende globale Suche nach $k = \frac{\varepsilon}{2 \cdot u_{\max}}$ Schritten durchgeführt werden.

Um die Inhalte der Halos zu finden, wird der in Abschnitt 4.3 beschriebene Algorithmus umgesetzt. Jedem virtuellen Slave wird ein Position-Code zugeordnet, über den die Positionssuche erfolgt. Die Codes werden in einem Feld gehalten, das bei wachsender Anzahl an Oberflächenknoten entsprechend vergrößert wird. Ein Feld erfüllt nicht die in der Analyse in Abschnitt 4.4 geforderte Effizienz bei der Sortierung der aktualisierten Codes. Die Anforderungen typischer Beispieleingaben haben jedoch gezeigt, dass der Grad der Vorsortierung hoch genug und diese pragmatischere Lösung ausreichend schnell ist. Für die Sortierung ist der *Insertion Sort*-Algorithmus gewählt, da dieser bei genügend guter Vorsortierung die wenigsten Vergleichs- und Vertauschoperationen

benötigt.

Zwischen den virtuellen Slaves und dem Feld der Position-Codes existiert eine bidirektionale Verbindung über Zeiger. Die Verbindung von den Slaves zu den entsprechenden Einträgen in dem Feld der Position-Codes wird nach der vollständigen Sortierung aktualisiert. Innerhalb der Applikation orientiert sich die Zellengröße an dem kürzesten Oberflächensegment im gesamten Netz. Die Anzahl der Zellen einer Dimension ist die kleinste Zweierpotenz, die für die Überdeckung des gesamten Berechnungsgebietes ausreichend ist.

5.3.4 Lokale Suche

Das wichtigste Ziel bei der Implementation der lokalen Kontaktsuche ist die Robustheit. Für jede denkbare Kontaktsituation soll die Kontaktbehandlung die Nichtpenetrationsbedingung erfüllen. Eine besondere Schwierigkeit ergibt sich aus numerischen Ungenauigkeiten, die bei der Berechnung der relativen Lage eines Slave zu einem Segment auftreten können. Die Lage (auf dem Segment oder links oder rechts des Segmentes) wird mittels des Skalarprodukts aus Segmentvektor und dem Verbindungsvektor vom Startpunkt des Segmentes zum Slave bestimmt. Aufgrund der endlichen Darstellung von Fließkommazahlen in Rechnern entspricht das Ergebnis des Vektorprodukts nicht unbedingt der tatsächlichen Lage.

Um die Problematik der numerischen Ungenauigkeit zu umgehen, ist das *Outside*-Flag implementiert. Es gibt an, ob ein Slave-Kandidat im letzten Zeitschritt auf der Materialseite oder auf der Nichtmaterialseite eines Oberflächensegmentes gelegen hat. Zum einen wird das Flag für ein notwendiges Kontaktkriterium benötigt, über das eine schnelle vorläufige Kontaktbestimmung möglich ist (vgl. folgenden Abschnitt). Zum anderen ermöglicht es auch dann eine *sinnvolle* Korrektur einer Penetration, wenn die geometrische Situation keine eindeutige Korrekturrichtung vorgibt. Ein Beispiel zum Outside-Flag ist in Abbildung 5.13 gegeben. An den Positionen B und C ist das Flag falsch, lediglich an Position A ist das Flag für den Slave-Kandidaten bzgl. des dargestellten Segmentes wahr.

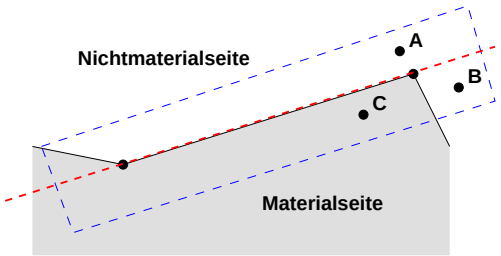


Abbildung 5.13: Outside-Flag: wahr für A, falsch für B und C

5.3.4.1 Vorläufiger Kontakttest

Die effiziente lokale Kontaktsuche läuft ihrerseits in zwei Phasen ab. In einer ersten Phase wird über ein notwendiges Kriterium entschieden, ob Kontakt zwischen einem Segment und seinem Kandidaten stattgefunden hat. Die zweite Phase führt für alle potentiellen Kontaktpaarungen einen exakten Kontakttest durch und bestimmt ggf. die Korrekturparameter.

In der ersten Phase der lokalen Kontaktsuche wird über einen einfachen Test festgestellt, ob an einem Slave-Kandidaten ein Kontakt stattgefunden haben kann. Aus diesen potentiellen Kontakten des aktuellen Zeitschritts werden die Aktivenlisten gebildet. Dazu wird das *Außerhalb-innerhalb*-Kriterium überprüft: Es kann nur dann ein initialer Kontakt stattfinden, wenn ein Kontaktkandidat die Gerade, die sich aus dem Oberflächensegment definiert, von Nichtmaterial- auf Materialseite überschritten hat. Dieser Test ist einfach durchzuführen und erlaubt damit eine schnelle und effektive Eingrenzung der potentiellen Kontaktpaare.

In der Abbildung 5.13 können die Knoten Kontakt zu dem Segment haben, die z. B. von Position A zu den Positionen B oder C gewechselt sind. Verbleibt ein Knoten im Bereich seiner Position, ist kein Eintrag in die Aktivenliste notwendig. Im Falle der Position A kann kein Kontakt stattgefunden haben. Knoten, die im Bereich der Position B liegen, können nur dann in das Material eindringen, wenn sie Kontakt zum Nachbarsegment haben. Ein Knoten an Position C hatte schon im letzten Schritt Kontakt und verbleibt für die folgenden Schritte in der Aktivenliste, bis bei der Bestimmung der Korrekturparameter die Beendigung der Kontaktsituation festgestellt wird.

Während der ersten Phase der Penetrationsbestimmung werden die Aktivengruppen gebildet. Der Algorithmus bearbeitet die Oberflächensegmente und deren Kandidatenlisten auf Basis der Kandidatengruppen, dem Ergebnis der globalen Kontaktsuche. Da die Knoten in den Aktivenlisten der Segmente immer eine Teilmenge der Knoten in den Kandidatenlisten darstellen, können die Gruppen lediglich kleiner werden oder in mehrere zerfallen, nicht jedoch verschmelzen.

5.3.4.2 Bestimmen der Korrekturparameter

In dieser Phase werden die Korrekturparameter (Richtung und Länge des Penetrationsvektors) bestimmt, auf deren Basis die Gleichungssysteme zur Kontaktkorrektur aufgestellt werden, je Kontaktgruppe ein System. Dazu ist zu gewährleisten, dass jeder Slave-Knoten innerhalb eines Gleichungssystems lediglich einfach auftritt, d. h. ein Slave darf innerhalb einer Kontaktgruppe nur einen Korrekturvektor haben und damit auf ein Oberflächensegment korrigiert werden. Die oben beschriebenen Phasen der Kontaktbehandlung ignorieren mögliche Konflikte. Ein Slave-Knoten kann sowohl in den Kandidatenlisten, als auch in den vorläufigen Aktivenlisten von benachbarten Oberflächensegmenten enthalten sein. In der gegebenen FE-Applikation ist die Halo-Größe jedoch so gewählt, dass sich Halos dreier benachbarter Oberflächensegmente nicht schneiden können. Ein Oberflächenknoten kann damit nur Kontaktkandidat von höchstens zwei

adjazenten Segmenten sein. Daraus ergibt sich der Vorteil, dass innerhalb der lokalen Kontaktsuche die Entscheidungsfindung zur Kontaktkorrektur auf jeweils zwei Segmente beschränkt ist.

In den meisten Fällen ist die Korrekturrichtung offensichtlich, in anderen Fällen hilft das Outside-Flag die geeignete zu finden. Insbesondere ist die Information des Outside-Flags behilflich, nichtphysikalische Beschleunigungen bei der Kontaktbehandlung zu vermeiden. Die Aufgabe des Outside-Flag bei der Korrekturphase ist in Abbildung 5.14 skizziert. Sie zeigt eine konvexe Masteroberfläche, in die ein Slave (roter Punkt) eingedrungen ist. Lag der Knoten im vorherigen Zeitschritt an Position A, führt die orthogonale Korrektur zum nächstliegenden Oberflächensegment zu einer Beschleunigung *in* Bewegungsrichtung, welche offensichtlich nichtphysikalisch ist. Unter der Einbeziehung des Outside-Flags kann ein Knoten jedoch nur in der Aktivenliste eines Segmentes liegen, wenn er im laufenden Zeitschritt von seiner Nichtmaterial- auf seine Materialseite gewechselt ist. Damit ist ein Slave, der aus dem Bereich um Position A kommt, immer nur in der Aktivenliste des linken Oberflächensegmentes. Damit ist die Korrekturrichtung von der vorherigen relativen Position zu den Oberflächensegmenten abhängig.

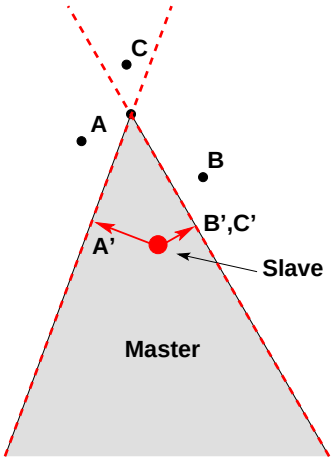


Abbildung 5.14: Korrekturvektoren (A',B',C') in Abhängigkeit von der Position im vorhergehenden Schritt der Zeitintegration (A, B, C)

Entscheidend für den Algorithmus zur Bestimmung der Korrekturrichtungen ist, dass die Entscheidung über eine Penetration lediglich über die vorläufigen Aktivenlisten der Oberflächensegmente getroffen wird. Zunächst wird überprüft, ob tatsächlich ein Kontakt stattgefunden hat. Dies ist der Fall, wenn der Slave-Knoten auch auf der Materialseite des Nachbarsegmentes liegt, das näher am untersuchten Slave-Knoten liegt. Ist der Slave auf die Nichtmaterialseite gewandert, aber nicht in das Material eingedrungen werden lediglich die Outside-Flags angepasst und der Knoten aus der Aktivenliste gelöscht.

Ist eine Kontaktsituation festgestellt, wird der eingedrungene Knoten wenn möglich orthogonal zur Oberfläche korrigiert. Ist der Slave nur zu einem Segment aktiv, ist die Korrekturrichtung eindeutig. Ergibt sich zu zwei benachbarten Segmenten die Möglich-

keit der orthogonalen Korrektur, wird das Segment mit der geringsten Eindringtiefe gewählt (vgl. Position C und Korrekturrichtung C' in Abb. 5.14). Im Falle konkaver Strukturen ist häufig keine orthogonale Korrektur möglich. Es wird eine Korrektur auf den mittleren Oberflächenknoten durchgeführt.

Ist ein eingedrungener Knoten zu einem Oberflächensegment aktiv, jedoch keine orthogonale Korrektur möglich und die nächstliegende Ecke an der Oberfläche konvex, bezeichnen wir ihn als *durchtauchenden Knoten*. Ein Knoten dringt nahe einer konvexen Ecke ein und bewegt sich gleichzeitig aus dem orthogonalen Korrekturbereich des Segmentes, in das er eingedrungen ist, heraus. In diesem Falle findet die Korrektur orthogonal zu dem Oberflächensegment statt, zu dem der Slave aktiv ist. Die Länge des Korrekturvektors ist so gewählt, dass der Knoten nur bis auf das benachbarte Oberflächensegment korrigiert wird. Die Situation ist in Abbildung 5.15 veranschaulicht. Ein Slave-Knoten, der im letzten Zeitschritt an Position A lag, ist in das obere Segment eingedrungen und liegt im aktuellen Zeitschritt an Position B. Die Korrektur findet orthogonal zum oberen Segment, jedoch nur bis auf das rechte Segment statt (roter Vektor). Eine Zerlegung in mehrere kleine Zeitschritte zeigt, dass diese Korrektur in etwa der Summe der Teilkorrekturen einer feingranulareren Simulation entspricht. Auch wenn in manchen Situationen dieser Lösungsansatz von der physikalischen Verhaltensweise abweicht, ist das Resultat die bestmögliche Lösung, falls die vorherige Position des Slaveknotens nicht bekannt ist.

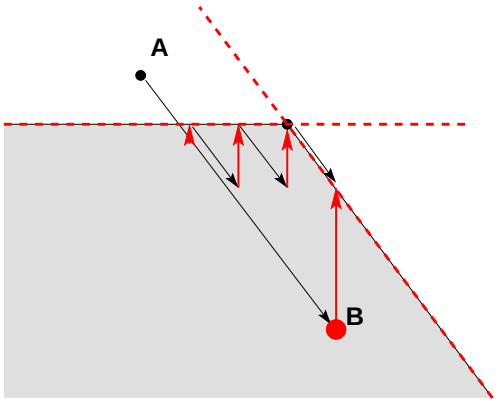


Abbildung 5.15: Korrektur im Falle eines *durchtauchenden Knotens*

Mit der Bestimmung des Penetrationsvektors ist gleichzeitig die zukünftige relative Lage von Master- und Slaveknoten zueinander bekannt. Diese Information wird genutzt, um die Outside-Flags auf beiden Seiten der Korrekturzone anzupassen und in den nachfolgenden Zeitschritten weiterhin eine vollständige und physikalische Korrektur zu gewährleisten. Damit werden Fehlentscheidungen aufgrund möglicher numerischer Ungenauigkeiten vermieden, die bei einer rein geometrischen Bestimmung der Outside-Flags getroffen werden können. Die vorliegenden Informationen erlauben die Aktualisierung von vier Segment-Knoten-Paarungen, im Umfeld des festgestellten Kontaktes: Der Slaveknoten mit dem aktuellen Mastersegment und dem näherliegenden Nachbarsegment und der näherliegende Masterknoten mit den am Slave angrenzenden Ober-

flächensegmenten. Ist eine der beiden Oberflächenstücke konkav, sind die Outside-Flags des gegenüberliegenden Knotens wahr. Ist die Oberfläche konvex, wird entsprechend der relativen Lage des Kontaktkandidaten das Flag für die eine Paarung auf wahr, für die andere auf falsch gesetzt.

Ein Beispiel, bei dem beide Seiten konvex sind, ist in Abbildung 5.16 gegeben. An den Knoten A und B grenzen die Segmente c und d bzw. e und f an. Der Slaveknoten B ist auf das Mastersegment d korrigiert worden. Die Outside-Flags der Paarungen (A, e) und (B, d) werden auf wahr, die der Paarungen (A, f) und (B, c) auf falsch gesetzt. Knoten A kann nur in das Segment f eindringen, wenn er zuvor auf die Nichtmaterialseite von Segment e gewechselt ist, ohne mit diesem in Kontakt zu treten. Vorher werden alle Kontakte des Knoten A auf das Segment e korrigiert. Das gleiche gilt für Knoten B und die beiden Segmente der anderen Komponente. Bei einer Korrektur im Bereich zweier konvexer Ecken findet immer diese Wertzuordnung an die Outside-Flags statt. Sie ist nur von dem Kontaktpaar (in diesem Fall (B, d)) abhängig. Die Winkel der Ecken und die Orientierung der Segmente ist dabei unerheblich.

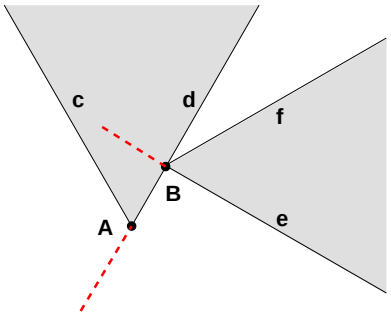


Abbildung 5.16: Aktualisierung des Outside-Flag: wahr für (A,e) und (B,d); falsch für (A,f) und (B,c)

Ein Sonderfall stellt die Korrektur eines Slaves auf einen Masterknoten dar. In diesem Fall werden alle vier betroffenen Outside-Flags auf wahr gesetzt. Beide Knoten können mit den beiden angrenzenden Segmenten der gegenüberliegenden Seite in Kontakt treten. Eine Korrektur findet dann auf das Segment statt, welches den geringsten Abstand zum eingedrungenen Knoten hat.

5.3.4.3 Geglättete Oberflächen

Mastersegmente an geglätteten Oberflächen benötigen keine Information bzgl. der relativen Lage ihrer Kontaktkandidaten. Aufgrund der Stetigkeit der Splines müssen keine speziellen Fälle betrachtet werden. Da die Glättung nur für den Bereich flacher Winkel der Netzoberfläche vorgesehen und die Halo-Größe klein genug gewählt ist, ist die orthogonale Korrektur immer möglich und auch eindeutig. Dennoch wird auch auf Mastersegmenten geglätteter Oberflächen zunächst ein vorläufiger Kontakttest durchgeführt, um vorläufige Aktivenlisten und zugehörige Aktivegruppen zu bilden. Dazu werden

die Oberflächen durch Linien approximiert, deren Überschreiten ein notwendiges Kontaktkriterium darstellen. Vor einem Kontakt mit einem konkaven Kurvensegment muss zunächst die Verbindungslinie der beiden Segmentmittelpunkte übertreten werden. Im Falle konvexer Kurven ist gegen die Verbindungslinie der Viertelpunkte zu testen (vgl. Abb. 5.17). Die beiden Linien stellen ein notwendiges Kriterium dar, da die Kurvensegmente immer innerhalb des Bereiches liegen, der von den beiden Verbindungslinien aufgespannt wird.

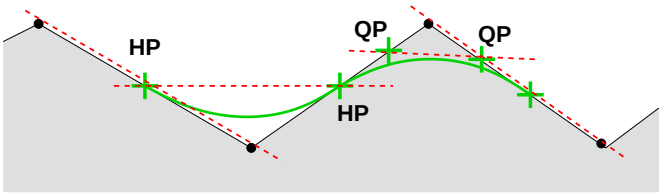


Abbildung 5.17: Vorläufiger Kontakttest an geglätteten Oberflächen

Die Analyse der Penetrationen arbeitet für alle Kombinationen aus eckigen und geglätteten Oberflächen. Der Algorithmus ist jeweils von der Art der Masteroberfläche abhängig. Auf der Slaveoberfläche werden nur Punkte betrachtet, welche entweder an den Oberflächenknoten des Netzes liegen (eckig) oder auf den Segmentmittelpunkten (geglättet). Zudem kann sich eine Aktivengruppen über beide Arten der Oberflächenstruktur erstrecken und somit alle Kombinationen innerhalb eines Gleichungsystems auftreten.

5.3.5 Benchmarks

Die FE-Netze zur Berechnung der gegebenen Problemstellungen eignen sich nur sehr eingeschränkt zur Untersuchung des Skalierungsverhaltens der beiden Position-Code-Algorithmen. In Netzen aus dem Bereich der Befestigungsproblematik ist die Anzahl der Oberflächensegmente gering gegenüber der Anzahl der Elemente. Damit führen große Oberflächen zu Testinstanzen mit einem sehr großen Speicheraufwand. Die Eingaben zur Risssimulation lassen sich nur schwer über eine feinere Diskretisierung skalieren, da sich das Verhalten bei der Simulation deutlich verändern kann. Zudem ist die Anzahl der Oberflächensegmente bei einer dynamischen Erweiterung nicht steuerbar. Daher werden in diesem Abschnitt synthetische Benchmark-Netze vorgestellt, deren Elemente und Oberflächensegmente gut skalierbar sind und deren Verhalten bei unterschiedlicher Elementanzahl in der Simulation vergleichbar bleibt.

Der Benchmark *Quadrate* simuliert ein System aus $n \times n$ Quadraten in einem umrandeten Gebiet. Die Größe der einelementigen Quadrate entspricht dem Abstand zwischen ihnen. Ein Viertel der elastischen Elemente im Innern werden mit Anfangsgeschwindigkeit 1 m/s in zufälliger Richtung gestartet. Es wird ein Zeitraum von 0.05 Sekunden mit Schrittweite $2 \cdot 10^{-6}$ Sekunden simuliert. Die Simulation erfolgt in allen Beispielen ohne Einbeziehung der Gravitation. Abbildung 5.18 zeigt die Instanz der Größe 16×16. Links ist die Startsituation, rechts die Positionen zum Ende der Simulation angegeben.

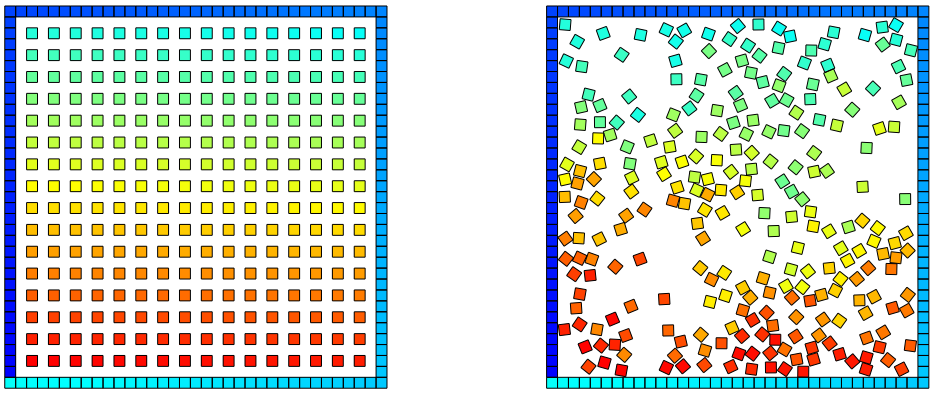


Abbildung 5.18: Benchmark *Quadrate* (16×16)

Die Farben sind gemäß der Elementnummer in der Eingabe gewählt und zeigen die Vermischung der Elemente während der Simulation.

Die Verhaltensweise des Benchmarks *Quadrate* liegt sehr nahe an dem Szenario, dass in der Analyse der algorithmischen Komplexität in Abschnitt 4.4 gewählt ist. Im gesamten Verlauf der Simulation sind die Knoten gleichmäßig verteilt und die Segmente nehmen beliebige Lagen im Raum an. Die Segmentlänge ist in dem Benchmark jedoch konstant.

In dem Benchmark *Stäbe* wird eine frei wählbare Anzahl an Stäben von einem Block zusammengedrückt. Die Stäbe sind 100 Elemente lang und ein Element breit. Der Raum zwischen den Stäben entspricht dem Doppelten ihrer Breite. Ein Block, der eine Anfangsgeschwindigkeit von 1 m/s hat, besitzt eine hohe Dichte, so dass das elastische Material der Stäbe weit zusammengedrückt wird. Das gesamte System ist wiederum durch eine Umrandung begrenzt. Die Instanz der Größe 8 ist in Abbildung 5.19 angegeben. Die untere Darstellung zeigt die Situation zum Ende der Simulation zum Zeitpunkt 0.06 Sekunden. Die Zeitschrittweite beträgt $2 \cdot 10^{-6}$ Sekunden.

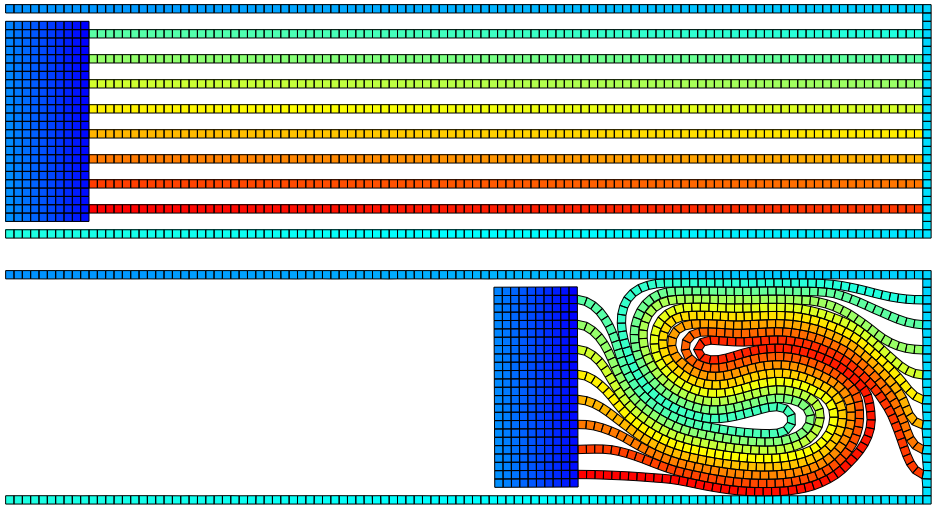


Abbildung 5.19: Benchmark *Stäbe* (8)

In dem Benchmark *Stäbe* ergeben sich deutlich größere Kontaktzonen. Die Anzahl der gefundenen Kontaktkandidaten steigt im Verlauf der Berechnung von 0.1 auf 1 je Oberflächensegment im Mittel. Beim Benchmark *Quadrate* liegt dieser Wert während des gesamten Verlaufs der Simulation bei 0.5. Der Benchmark spiegelt die Anforderungen bei der Simulation von Befestigungsvorgängen wider, bei dem große Kontaktzonen zwischen Nagel und Untergrund und zwischen zu befestigenden Blechen und dem Untergrund auftreten können.

Der Benchmark *Ringe* beinhaltet eine frei wählbare Anzahl an Ringen, wobei im Folgenden der gelagerte äußere Ring nicht in die Bestimmung der Größe einfließt. Der innere Ring hat einen Innenradius von 1 mm und besteht aus 36 Elementen. Nach außen wächst die Anzahl jeden zweiten Ring um ein Element. Der Abstand zwischen den Ringen beginnt bei 0.1 mm und steigt um jeweils sechs Prozent. Die Breite der Ringe orientiert sich an der resultierenden Länge der inneren Oberflächensegmente, was etwa 0.175 mm im ersten Ring bedeutet. Alle inneren Ringe erhalten eine Anfangsgeschwindigkeit von 1 m/s. Der erste wird in y-Richtung gestartet, die weiteren in jeweils 135° gedrehter Richtung. Die Simulation erfolgt über 0.01 Sekunden mit einer Schrittweite von 10^{-6} Sekunden.

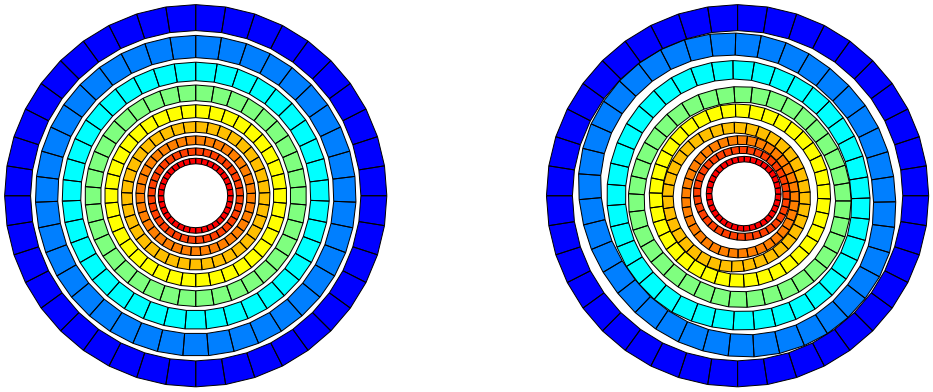


Abbildung 5.20: Benchmark *Ringe* (8)

Die große Anzahl an Segmenten mit kleinem Abstand zueinander ergibt sich vielfach bei der Simulation von Rissbildungen in sprödem Material (vgl. Abb. 5.4 (Zylinder), S. 106). Im Gegensatz zu den anderen synthetischen Szenarien ist das Netz in diesem Falle lokal verfeinert. Die Segmente des 44sten Ringes sind z. B. um den Faktor acht größer als die des ersten. Insbesondere bei der Simulation der Abbauvorgänge ergeben sich diese enormen Unterschiede, da die Größe des Untergrundes im Allgemeinen keine gleichmäßig feine Diskretisierung erlaubt. Diese Problematik ergibt sich auch in vielen anderen Anwendungsgebieten der FE-Simulation.

5.3.6 Ergebnisse

Das Verhalten der beiden Varianten des Position-Code-Algorithmus (vgl. Abschnitt 4.3) innerhalb der globalen Kontaktsuche wird anhand der durchgeführten Vergleichsopera-

tionen untersucht. Dazu sind die Programmteile über Zählvariablen instrumentiert, die jeden Vergleich im Quellcode des Programms aufsummieren. Diese Metrik ermöglicht eine Konzentration auf die algorithmischen Eigenschaften der Verfahren. Ein Vergleich der Laufzeiten der Kontaktsuche ergibt eine komplexe Abhängigkeit mit der tatsächlichen Umsetzung der Algorithmen, dem Verhalten des Compilers und der Zielmaschine. Zudem steht an dieser Stelle die Untersuchung des Skalierungsverhalten der beiden Verfahren im Vordergrund, nicht die tatsächliche Leistung im Einzelfall. Die algorithmische Analyse hat gezeigt, dass die Konstanten bei der Variante über raumfüllende Kurven deutlich höher liegen. Damit ist offensichtlich, dass die einfachere Variante über die linearen Position-Codes bei kleinen Eingabeinstanzen zu schnelleren Laufzeiten führt.

In Abbildung 5.21 sind die Ergebnisse zum Benchmark *Quadrate* dargestellt. Während der 25 000 Schritte dauernden Simulation wird die globale Kontaktsuche 359-mal bis 696-mal durchgeführt. Die Varianz in der Häufigkeit liegt an den unterschiedlichen Geschwindigkeiten die während der Simulation auftreten können. Je mehr Quadrate sich beliebig im Raum bewegen, desto größer ist die Wahrscheinlichkeit, dass ein Knoten eine hohe Geschwindigkeit erreicht. In den Abbildungen sind die ausgeführten Vergleichsoperationen je Oberflächensegment und je Kontaktsuche dargestellt.

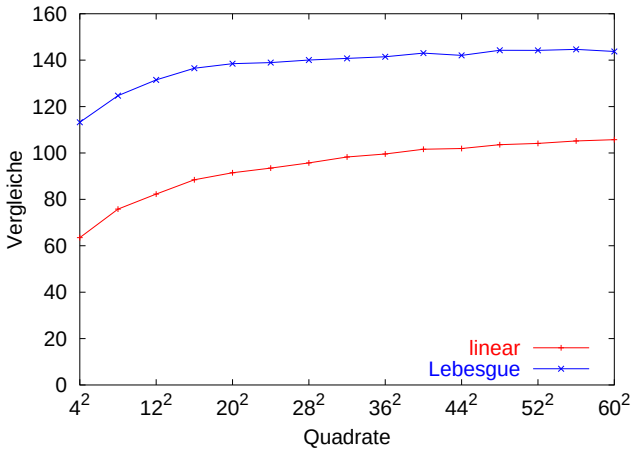


Abbildung 5.21: Mittlere Kosten der globalen Kontaktsuche für den Benchmark *Quadrate*

Abgesehen von den kleineren Instanzen liegt die gesamte Anzahl an Vergleichen für die Variante des Position-Code-Algorithmus basierend auf der Lebesgue-Kurve konstant bei etwa 145 Vergleichen. Der Verlauf der Messpunkte für die lineare Variante steigt im gesamten untersuchten Bereich. Der Overhead für die raumfüllende Kurve sinkt daher von 52 auf 37 Vergleiche.

Wie sich die Aufwände auf die einzelnen Phasen des Algorithmus verteilen ist in Abbildung 5.22 und 5.23 dargestellt. Die Diagramme zeigen die normierten Vergleiche für die Phasen *Aktualisieren*, *Sortieren*, *Suchen* und *Testen*. Die einzelnen Phasen lassen sich für das Beispiel *Quadrate* gut mit den analytisch ermittelten Werten aus Abschnitt 4.4 vergleichen (vgl. Tab. 4.5, S. 90).

Die Anzahl der Vergleiche für das Aktualisieren der Position-Codes entspricht dem in

der Analyse ermittelten minimalen Aufwand. Während er bei der linearen Variante offensichtlich konstant ist, fällt der ermittelte variable Anteil von fünf Vergleichen je verschobenem Knoten im Mittel in diesem Beispiel nicht ins Gewicht. Bei jeder globalen Kontaktsuche sind nur sehr wenige Knoten in eine benachbarte Zelle gewandert.

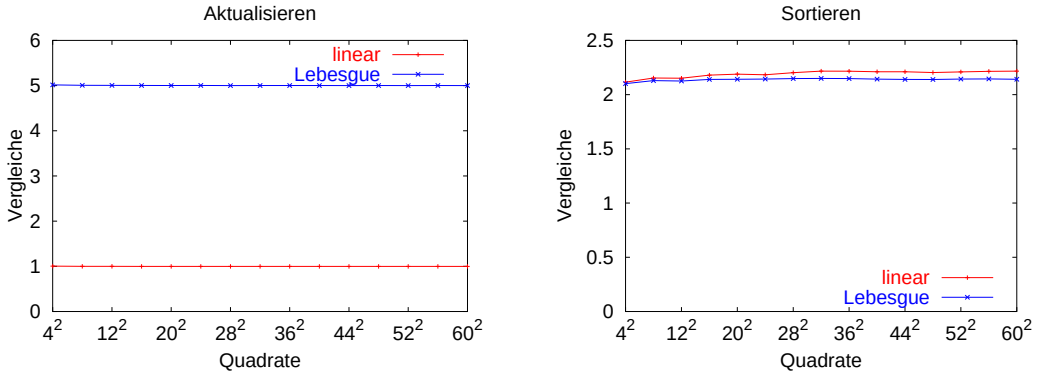


Abbildung 5.22: Mittlere Kosten der Phasen *Aktualisieren* und *Sortieren* für den Benchmark *Quadrate*

Ähnliches gilt für die Sortierphase in beiden Varianten. Der variable Anteil von $8 \log_2 n + 40$ (linear) bzw. 80 (Lebesgue-Kurve) Vergleichen im Mittel gilt nur für Knoten, deren Position sich in der sortierten Liste verändert hat. Es lässt sich ein leichter Anstieg für die lineare Variante erkennen, der von dem logarithmischen Term hervorgerufen wird. Dagegen bleibt die Anzahl der Vergleiche bei der Lebesgue-Kurve weitgehend konstant.

Ein deutlicherer Unterschied zeigt sich für das Finden der Suchintervalle innerhalb der Liste in Abbildung 5.23 links. Die Messwerte der Lebesgue-Variante konvergieren gegen einen Wert von etwa 70 Vergleichen, welcher dem Ergebnis der Analyse des Verfahrens entspricht. Der logarithmische Anteil ($\frac{8s+8}{\pi} \log_2 n$) des linearen Position-Code-Algorithmus hat in diesem Fall einen deutlich größeren Einfluss auf die Gesamtkosten, da er von der Bewegung der Knoten unabhängig ist. Die Analyse hat ergeben, dass 11 Vergleiche unabhängig von der Größe der Eingabeinstanz sind. Der größenabhängige Anteil beträgt in diesem Fall 25 bis 55 Vergleiche.

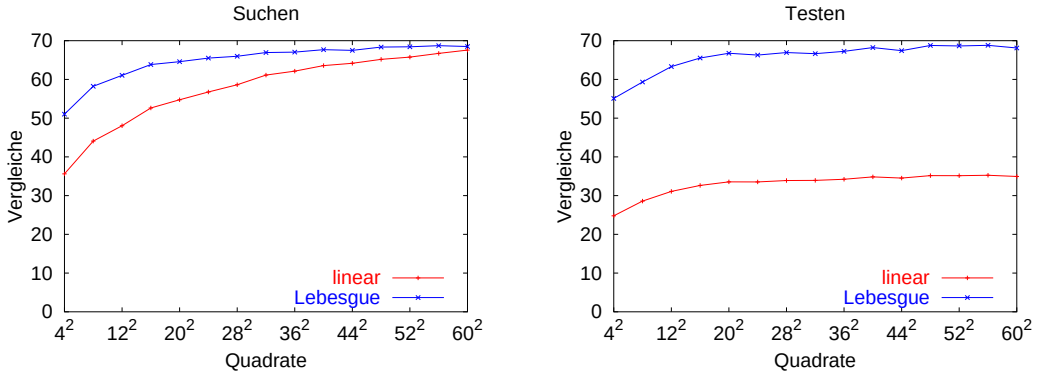


Abbildung 5.23: Mittlere Kosten der Phasen *Suchen* und *Testen* für den Benchmark *Quadrate*

Das Testen der Knoten in den Suchintervallen gegen den Halo ist in Abbildung 5.23 rechts dargestellt. Die Messwerte konvergieren gegen 35 (linear) und 70 (Lebesgue) Vergleiche.

Der Anstieg der Vergleiche in den Phasen *Suchen* und *Testen* wird durch den geringer werdenden Anteil des Randes am gesamten FE-Netz hervorgerufen. An der Außenseite des Randes werden keine Kontaktkandidaten im Halo liegen und auch nur wenige in den Suchintervallen. Zudem sind die Kosten für das Finden dieser Intervalle am oberen und unteren Rand deutlich geringer als im Mittel.

Der logarithmische Anteil in den Phasen *Sortieren* und *Suchen* bei der linearen Variante des Position-Code-Algorithmus ist so gering, dass er durch die allgemein geringeren Konstanten in allen Phasen mehr als kompensiert wird. Bei dem untersuchten Szenario ist eine sehr große Instanz notwendig, für die die absoluten Kosten (Anzahl der Vergleiche) der Lebesgue-Kurve geringer sind, als bei der linearen Sortierung. Rechnerisch liegt die Größe der Instanz bei etwa 800×800 Quadraten.

Abbildung 5.24 zeigt die Ergebnisse zum Benchmark *Stäbe*. Die Messpunkte linear x und Lebesgue x zeigen die durchgeführten Vergleiche in den untersuchten Instanzen zu dem in Abbildung 5.19 beschriebenen Szenario. Es ergibt sich eine Differenz von etwa 27 Vergleichen über alle Eingabegrößen. Die in der Analyse aufgezeigten logarithmischen Terme bei der linearen Algorithmus-Variante ergeben sich aus der erwarteten Höhe der Segmente bzw. die Höhe der sich aus diesen Segmenten ergebenden Halos. Da in dem Benchmark-Netz die Oberflächensegmente während der Simulation zu einem großen Teil horizontal orientiert sind, ergibt die Analyse eines solchen Szenarios konstante Kosten für beide Varianten des Position-Code-Algorithmus. Der leicht fallende Verlauf der Messkurven erklärt sich durch die leicht unterschiedlichen Verhaltensweisen der Simulationen für verschiedene Instanzen. Aufgrund von Schwingungen im Material ergeben sich sehr kurze Abstände zwischen zwei globalen Kontaktsuchphasen, die im Mittel von 7.3 bei der kleinsten Instanz auf 4.8 in der größten sinken.

Die Aufteilung der Kosten auf die einzelnen Phasen ist mit dem obigen Benchmark

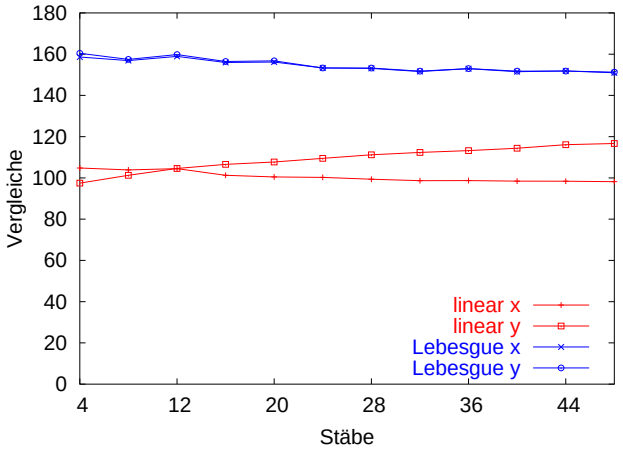


Abbildung 5.24: Mittlere Kosten der globalen Kontaktsuche für den Benchmark *Stäbe*

vergleichbar. Die Kosten der Sortierphase liegen bei konstant einem bzw. fünf Vergleichen je Knoten und globaler Kontaktsuche. Für das Sortieren werden etwas mehr als zwei Vergleiche im Mittel benötigt. Für das Suchen der zu testenden Knotenintervalle innerhalb der Position-Code-Liste werden 60 bis 70 (linear) bzw. 70 bis 80 (Lebesgue) Vergleiche benötigt. Die Vergleiche in der Testphase liegen mit 70 bis 80 für die Lebesgue-Variante etwa doppelt so hoch wie die des linearen Ansatzes.

Der Einfluss der Lage der Oberflächensegmente im Raum auf die Komplexität der beiden Verfahren zeigt sich bei einer Drehung des Netzes um 90 Grad. Die Ergebnisse zu dem modifizierten Benchmark sind in Abbildung 5.24 als linear y und Lebesgue y angegeben. Es zeigt sich, dass die Lebesgue-Variante unabhängig von der Lage der Segmente ist. In beiden Fällen ergeben sich nahezu identische Anzahlen an Vergleichsoperationen. Dagegen zeigt sich bei der linearen Variante ein deutlicher Anstieg der Kosten bei wachsenden Instanzen, wenn ein Großteil der Segmente vertikal angeordnet ist. Der Overhead durch die höheren Konstanten bei der Lebesgue-Variante sinkt in dem untersuchten Bereich von 62 auf 36 Vergleiche je Segment und Kontaktsuchphase. Unter Berücksichtigung des analytisch ermittelten asymptotischen Verhaltens wird der Schnitt der beiden Messkurven bei etwa 150 Stäben erfolgen. Damit ist wie im obigen Benchmark eine sehr große Eingabeinstanz von fast 25 000 Elementen erforderlich.

Der Benchmark *Ringe* zeigt die Abhängigkeit der algorithmischen Komplexität von der Länge der Segmente, zu denen die Kontaktkandidaten gefunden werden müssen. Während der 10 000 Schritte dauernden Simulation werden 767 (4 Ringe) bis 1156 (44 Ringe) globale Kontaktsuchen durchgeführt. Abbildung 5.25 zeigt die normierte Anzahl der Vergleiche für die beiden untersuchten Varianten des Position-Code-Algorithmus.

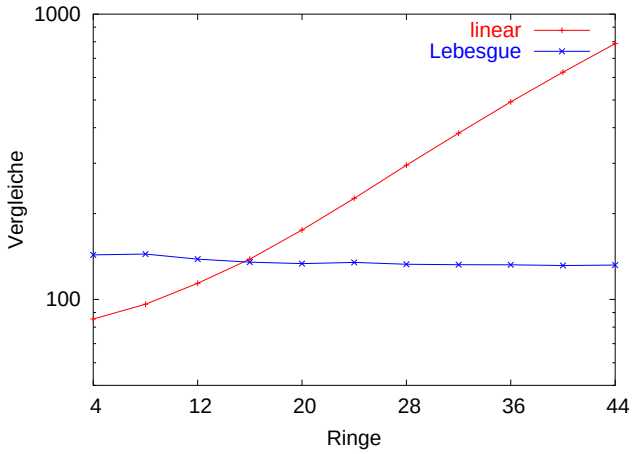


Abbildung 5.25: Mittlere Kosten der globalen Kontaktsuche für den Benchmark *Ringe* (logarithmische Darstellung)

Für die Lebesgue-Kurve als Basis der Position-Codes ergibt sich ein leicht fallender Verlauf der Messwerte von 144 auf 132 Vergleiche im Mittel. Die Kosten der einzelnen Phasen sind mit Ausnahme des Halo-Tests mit der analytisch ermittelten Komplexität vergleichbar. Für das Aktualisieren der Codes sind im Mittel fünf, für das Sortieren

knapp über zwei Vergleiche erforderlich, da die Bewegung zwischen zwei globalen Kontaktsuchen auch in diesem Beispiel recht gering ist.

Die Kosten für das Suchen der relevanten Knotenintervalle erfordert etwa 70 Vergleiche und liegt damit in der gleichen Größenordnung wie bei den anderen Benchmarks. Ab 16 Ringen ergibt sich ein leichter Anstieg, der die Auswirkung des logarithmischen Terms des Analyseergebnisses ($33 \log_2 s$) widerspiegelt. Die längeren Segmente machen jedoch nur einen kleineren Teil der Gesamtoberfläche aus. Zudem gilt die Annahme einer gleichmäßigen Verteilung der Knoten im Berechnungsgebiet nicht mehr, so dass die Abstände zwischen den Knotenintervallen mitunter deutlich geringer sind, als bei dem zur Analyse gewählten Szenario. Die resultierenden zusätzlichen Kosten sind im Gesamtmittel gering und ohne nennenswerten Einfluss auf die gesamte globale Kontaktsuchphase.

Beim Testen der gefunden Knoten gegen die Halos der einzelnen Segmente ergibt sich ein deutlicherer Unterschied zu den analytisch ermittelten Kosten. Die Verteilung der Knoten ist im Gegensatz zu dem analytisch untersuchten Szenario nicht gleichmäßig sondern von geringerer Dichte im Bereich längerer Segmente. Daher bleibt die Zahl der Knoten in Gebieten, die gegen den Halo getestet werden müssen, konstant, an Stelle des angenommenen quadratischen Wachstums gegenüber der Segmentlänge. Der Aufwand fällt von 67 auf 56 Vergleiche bei wachsender Instanz. Insgesamt zeigt sich, dass der Position-Code-Algorithmus basierend auf der Lebesgue-Kurve auch für adaptiv diskretisierte Netze einen im Mittel nahezu konstanten Aufwand für unterschiedliche Netzgrößen hat.

Die Messwerte des linearen Position-Code-Algorithmus zeigen einen exponentiellen Anstieg für die Kosten der globalen Kontaktsuche. Dieser Anstieg ergibt sich aus der Ineffizienz der Suchphase. Die Analyse des Verfahrens in Abschnitt 4.4 hat gezeigt, dass die Suchphase von der Länge des Oberflächensegmentes abhängig ist. In dem gewählten Szenario liegt diese Abhängigkeit bei $\frac{8s+8}{\pi} \log_2 n$. Das Aktualisieren und Sortieren der Knoten verhält sich vergleichbar zu den vorherigen Benchmarks und damit auch zu den Ergebnissen der Kostenanalyse im vorgestellten Szenario. Für das Testen der Knoten gegen den Halos der einzelnen Oberflächensegmente ergibt sich, wie bei der Lebesgue-Variante, durch die nach außen dünnere Verteilung von den Segmentelängen unabhängige Kosten von 22 bis 26 Vergleichen im Mittel.

Kapitel 6

Zusammenfassung

In dieser Arbeit wurden diskrete raumfüllende Kurven in zwei Anwendungsgebieten, der Graphpartitionierung und der Kontaktsuche, untersucht. Die Graphpartitionierung ist eine zentrale Aufgabe bei der Parallelisierung von Simulationen nach der Finite-Elemente-Methode. Die Rechenlasten müssen gleichmäßig auf die Recheneinheiten verteilt und der Kommunikationsaufwand minimiert werden. Die Kontaktsuche ist eine bedeutende Phase bei der Simulation strukturmechanischer Prozesse. In vielen Applikationen beansprucht sie einen wesentlichen Anteil an der Gesamtrechenzeit. Für beide Bereiche sind analytische und experimentelle Ergebnisse für die Bewertung der Verfahren auf Basis raumfüllender Kurven erarbeitet worden.

Die Partitionierung über raumfüllende Kurven, die zur Klasse der geometrischen Verfahren zählt, ist schon in mehreren parallelen Anwendungen eingesetzt worden. In dieser Arbeit ist die Qualität der erzielten Partitionierungen erstmalig detailliert analysiert und mit den Lösungen eines hochwertigen mehrstufigen Verfahrens verglichen worden. Für Gitter oder gitterähnliche Netze ergibt sich ein moderater Anstieg im Kantenschnitt. Dahingegen ergibt sich bei unstrukturierten Netzen ein stärkerer Qualitätsverlust, insbesondere bei der Partitionierung in wenige Teile. Die Analyse über die vorgeschlagene Normierung des Kantenschnitts zeigt, dass die untersuchten unstrukturierten Netze eine Partitionierung höherer Qualität erlauben, die von dem mehrstufigen Verfahren gefunden werden. Dahingegen ist die von raumfüllenden Kurven erzielte Qualität i. A. unabhängig von der Struktur der Netze.

Die besonderen Vorteile der Indizierung der Knoten des Graphen über die Struktur einer raumfüllenden Kurve ergeben sich in der Möglichkeit der impliziten Partitionierung. Für jede gewünschte Anzahl an Partitionen und jede gegebene Gewichtung der Knoten folgen direkt die gesuchten Partitionen in Form von Intervallen innerhalb der geordneten Knotenfolge. Damit stehen dem erhöhten Aufwand für die Kommunikation aufgrund des höheren Kantenschnitts erheblich reduzierte Kosten für die Bestimmung der Repartitionierung gegenüber. Die implizite Partitionierung wird dann vorteilhafter sein, wenn die Applikation eine häufige Lastbalancierung erfordert.

Für die globale Kontaktsuche wurde eine neue Variante einer zellbasierten Methode

vorgeschlagen, die der Ordnung über raumfüllende Kurven folgt. Die Analyse des algorithmsichen Aufwands zeigt, dass diese Variante asymptotisch geringe Kosten erfordert. Bei Netzen moderater Größe ergibt sich immer dann ein Vorteil gegenüber der ursprünglichen Methode, wenn es eine deutliche Varianz in der Diskretisierung des gegebenen Netzes gibt.

Für beide Anwendungsfelder sind verschiedene Kurven analytisch untersucht und verglichen worden. Neben einem bestehenden Qualitätsmaß für die Partitionierungseigenschaften wurde ein neues Maß für die Eignung innerhalb der globalen Kontaktsuche definiert. In beiden Fällen konnten Schranken zum Verhalten der Kurven im zweidimensionalen Gitter bestimmt werden. Die wichtigsten Ergebnisse zur Partitionierungsqualität sind die genauen Schranken für die Lebesgue-Kurve im mittleren Fall und die Tatsache, dass die Lebesgue-Kurve die beste Qualität aller untersuchten Kurven im schlechtesten Fall erzielt. Desweiteren hat eine umfassende experimentelle Analyse gezeigt, dass die neu vorgestellte $\beta\Omega$-Kurve im mittleren Fall die besten Partitionierungen liefert. Für die logarithmische Index-Distanz, dem Qualitätsmaß für die Eignung zur globalen Kontaktsuche, konnte gezeigt werden, dass im mittleren Fall die $\beta\Omega$-Kurve die beste Lokalität erzielt, gefolgt von der Hilbert- und der Lebesgue-Kurve.

Die vorgestellten Ansätze zur Partitionierung und Kontaktsuche sind in einer industriellen Applikation integriert worden. In diesem Umfeld konnte die Eignung raumfüllender Kurven für die entsprechenden Aufgabenstellungen gezeigt werden.

Teile der in dieser Arbeit gezeigten Ergebnisse wurden bereits auf internationalen Konferenzen präsentiert. Vorläufige Ergebnisse zur Partitionierungsqualität raumfüllender Kurven sind auf der *International Conference on Computational Science 2002* vorgestellt worden [39]. Die Untersuchungen zur Eignung raumfüllender Kurven für die Partitionierung irregulärer Graphen sind gemeinsam mit Stefan Schamberger auf der *Parallel Computing Technologies 2003* Konferenz veröffentlicht [72]. Zusammen mit Hans-Peter Gänser und Jan Hungershöfer sind die Ergebnisse zur impliziten Partitionierung innerhalb der industriellen Applikation auf der *International Conference on Compuational Science and its Applications 2003* präsentiert worden [40].

Die analytischen Ergebnisse zu den Eigenschaften der raumfüllenden Kurven in der globalen Kontaktsuche sind auf der *Canadian Conference on Computational Geometry 2002* vorgestellt worden [82]. Der Algorithmus zur Kontaktsuche über raumfüllende Kurven und die Ergebnisse innerhalb der industriellen Applikation sind gemeinsam mit Ralf Diekmann, Jan Hungershöfer, Michael Lux und Lars Taenzer auf dem *European Congress on Computational Medhods in Applied Sciences and Engineering 2000* und dem *IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation* präsentiert worden [19, 20].

Anhang A

Partitionierungsqualität der Lebesgue-Kurve

In Abschnitt 3.3.3 ist ein Beweis für die obere Schranke der mittleren Partitionierungsqualität der Lebesgue-Kurve,

$$\mathcal{P}_{\text{avg}}^{\text{Lebesgue}} \leq \frac{5}{2\sqrt{3}} \ ,$$

angegeben. Die Beweisidee basiert auf der bekannten Wahrscheinlichkeit, mit der eine benachbarte Zelle im Gitter zu der gleichen Partition gehört. Hier ist ein weiterer Beweisansatz angegeben. Es werden alle möglichen Fälle aufgezählt, in denen eine direkt benachbarte Zelle zur selben Partition gehört. Damit ergibt sich auch ein Term, der den mittleren Rand einer Partition im *endlichen* Gitter exakt beschreibt.

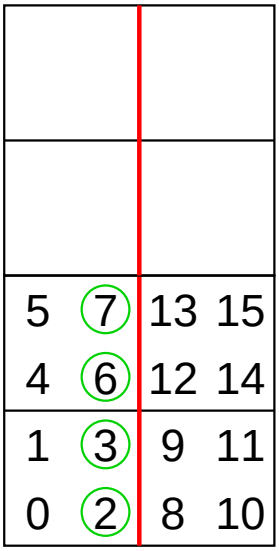
Lemma A.1 *Für die Oberfläche S einer Partition der Größe V, die durch die Lebesgue-Kurve definiert ist, gilt im mittleren Fall*

$$\begin{aligned} S \leq \ & \tfrac{3}{2^k}V + \tfrac{8}{3}2^k - \tfrac{5}{3}\tfrac{1}{2^k} \quad \text{für } V \in \left[\frac{2\cdot 4^k+1}{3}, \frac{4\cdot 4^k+2}{3}\right[\text{ und} \\ S \leq \ & \tfrac{2}{2^k}V + 4\cdot 2^k - \tfrac{1}{2^k} \quad \text{für } V \in \left[\frac{4\cdot 4^k+2}{3}, \frac{2\cdot 4^{k+1}+1}{3}\right[, \text{mit } k \in \mathbb{N}_0 \ . \end{aligned}$$

Beweis: Sei $N = n \times n$ die Größe des gegebenen Gitters und $L = \log_2 n$ die Anzahl der Level der Lebesgue-Kurve. Betrachte eine Partition der Größe V und ein vertikalen Separator vom Level l. Für die Bestimmung der Anzahl aller inneren Kanten wird eine Region der Größe $2^{2(l+1)}$ untersucht.

Beispiel A.1 Abbildung A.1 zeigt die entsprechende Region für $l = 1$ und die Ordnung der Lebesgue-Kurve innerhalb des Bereiches. Die Tabelle gibt die Abhängigkeit an, die zwischen der Partitionsgröße, der Position der ersten Zelle der Partition und der Anzahl innerer Kanten am vertikalen Separator vom ersten Level (rote Linie im linken Bild) existieren. Die linke Spalte gibt die erste Zelle der Partition an und zwar als relative Position bzgl. des vertikalen Separators vom Level 1 (vgl. linke Darstellung).

In Abhängigkeit von dieser Position ist das minimale Volumen angegeben, das benötigt wird, um eine, zwei, drei oder vier innere Kanten vom Level 1 innerhalb der Partition zu haben. Ist vier die erste Zelle der Partition, dann muss die Partition aus mindestens 21 Zellen bestehen, damit drei innere vertikale Kanten vom ersten Level enthalten sind.



i \ j	1.	2.	3.	4.
0	9	10	13	14
1	8	9	12	13
2	7	8	11	12
3	7	10	11	22
4	9	10	21	22
5	8	9	20	21
6	7	8	19	20
7	7	18	19	22
8	17	18	21	22
9	16	17	20	21
10	15	16	19	20
11	14	15	18	19
12	13	14	17	18
13	12	13	16	17
14	11	12	15	16
15	10	11	14	15

Abbildung A.1: Innere vertikale Kanten am Separator vom Level 1

Im Folgenden sind die Werte der Tabelle definiert als $\delta(i, j)$, mit Zeilennummer i (erste Zelle) und Spalte j (Anzahl innerer Kanten an diesem Level)

Für die Zellen, die links neben dem betrachten Separator liegen (grün markiert) gilt $\delta(i, 1) = 7$ $(= R_l + 1 = \frac{4 \cdot 4^l + 2}{3} + 1$, genau eine Zelle mehr als die Länge des Pfades der Lebesgue-Kurve zwischen den beiden Zellen). Desweiteren gilt für diese Zellen auch $\delta(i, j) = \delta((i + 1) \bmod 16, j - 1) + 1$. Die minimale Partitionsgröße für die j-te innere Kante ist um eins größer, als die minimale Partitionsgröße der $(j-1)$-ten inneren Kante der folgenden Zelle. Für alle anderen Zellen gilt $\delta(i, j) = \delta((i + 1) \bmod 16, j) + 1$. Die minimale Partitionsgröße ist direkt vom Wert der folgenden Zelle abhängig.

Die Formulierung von δ ist nur für beschränkte Partitionsgrößen definiert. Eine Erweiterung für beliebige Partitionsgrößen ist

$$\gamma(V, l, i, j) = \left\lfloor \frac{V + 4^{(l+1)} - \delta(i, j)}{4^{(l+1)}} \right\rfloor .$$

γ bezeichnet die Anzahl an Blöcken, in denen wir j innere Kanten erwarten können, wiederum in Abhängigkeit der relativen Position i der ersten Zelle.

Die Häufigkeit, mit der eine relative Position i die erste Zelle einer Partition ist, ist gegeben durch

$$\varphi(L,V,l,i) = \left\lfloor \frac{4^L - V + 4^{(l+1)} - i}{4^{(l+1)}} \right\rfloor \ .$$

Sie hängt vom Volumen der Partition und der Größe des gegebenen Gitters ab, definiert durch die Rekursionstiefe L der Lebesgue-Kurve.

Aus diesem folgt für die mittlere Anzahl E der inneren Kanten am vertikalen Separator vom Level l,

$$E_v(l) = \frac{\displaystyle\sum_{i=0}^{2^{2(l+1)}} \left(\varphi(L,V,l,i) \cdot \sum_{j=1}^{2^{l+1}} \gamma(V,l,i,j) \right)}{\displaystyle\sum_{i=0}^{2^{2(l+1)}} \varphi(L,V,l,i)} \ .$$

In δ fehlt noch die Definition der speziellen Zellen, die direkt am Separator vom Level l liegen (vgl. grüne Markierung in Abb. A.1). Diese Positionen werden im Folgenden als i_1, i_2, ... bezeichnet. Zunächst betrachten wird den Abstand zwischen diesen Zellen: $\Delta_k = i_{k+1} - i_k$. Wegen der regelmäßigen Struktur der Lebesgue-Kurve, lässt er sich leicht generieren:

$$\begin{aligned} \Delta_k &= (1,3,1,11,1,3,1,43,1,3,1,11,\ldots) \\ &= \frac{2 \cdot 4^i + 1}{3} \ , \text{ für } k = 2^i + j \cdot 2^{i+1} \text{ mit } i,j \in \mathbb{N}_0 \ . \end{aligned}$$

Das heißt, der Wert $\frac{2 \cdot 4^i + 1}{3}$ steht innerhalb der Folge zum ersten Mal an Position 2^i und wiederholt sich an jeder 2^{i+1}-ten Position.

Die relative Position der ersten Zelle (i_1), welche von dem untersuchten Level l abhängt, ist

$$\Delta_0 = \frac{2 \cdot 4^l - 2}{3} \ .$$

Die Definition von δ zeigt, dass die Werte von ihrer Distanz zur j-ten speziellen Zelle abhängen. Diese Distanzen sind ebenfalls in der Folge Δ_k zu finden. Für die weitere Berechnung ist es notwendig, die Aufzählung aller Fälle in die Intervalle aufzuteilen, die sich aus den speziellen Zellen ergeben. Seien $i_I = \displaystyle\sum_{k=0}^{I-1} \Delta_k$ die relativen Positionen der speziellen Zellen. Es ergeben sich drei Teilsummen für die Intervallgruppen $i \leq i_1$, $i_I < i \leq i_{I+1}$ mit $I \in [1, 2^{l+1}[$ und $i_{2^{l+1}} < i$. Zur Vereinfachung gelte im Folgenden $\Phi(L,V,l) = \displaystyle\sum_{i=0}^{2^{2(l+1)}} \varphi(L,V,l,i)$. Es gilt

$$\begin{aligned} E_v(l) &= \sum_{i=0}^{i_1} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma(V,l,i,j) \right) \\ &\quad + \sum_{I=1}^{2^{(l+1)}-1} \sum_{i=i_I+1}^{i_{(I+1)}} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma(V,l,i,j) \right) \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{i=i_{2^{2(l+1)}}+1}^{2^{2(l+1)}} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma(V,l,i,j) \right) \\
= &\sum_{i=0}^{i_1} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma_1(V,l,I,i,j) \right) \\
&+ \sum_{I=1}^{2^{(l+1)}-1} \sum_{i=i_I+1}^{i_{(I+1)}} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma_2(V,l,I,i,j) \right) \\
&+ \sum_{i=i_{2^{2(l+1)}}+1}^{2^{2(l+1)}} \left(\frac{\varphi(L,V,l,i)}{\Phi(L,V,l)} \cdot \sum_{j=1}^{2^{l+1}} \gamma_3(V,l,I,i,j) \right)
\end{aligned}
$$

mit

$$
\begin{aligned}
\gamma_1(V,l,I,i,j) &= \left\lfloor \frac{V + 4^{(l+1)} - (R_l + i_j - i)}{4^{(l+1)}} \right\rfloor \\
\gamma_2(V,l,I,i,j) &= \left\lfloor \frac{V + 4^{(l+1)} - \left(R_l + (i_{(j+I) \bmod 2^{l+1}} - i) \bmod 4^{(l+1)} \right)}{4^{(l+1)}} \right\rfloor \\
&= \left\lfloor \frac{V + 4^{(l+1)} - \left(R_l + i_{(j+I)} - i \right)}{4^{(l+1)}} \right\rfloor \\
\gamma_3(V,l,I,i,j) &= \left\lfloor \frac{V + 4^{(l+1)} - \left(R_l + i_j + 4^{(l+1)} - i \right)}{4^{(l+1)}} \right\rfloor
\end{aligned}
$$

und

$$
R_l = \frac{4 \cdot 4^l + 2}{3} \; .
$$

Dabei ist R_l die Distanz über den Pfad der Kurve zum direkten Nachbarn an einem vertikalen Separator vom Level l.

Die exakte Formulierung von $E_v(l)$ erlaubt keine direkte Vereinfachung. Für sehr große Gitter gilt jedoch, dass $\varphi(L,V,l,i)$ gegen $\varphi(L,V,l,0)$ konvergiert. Das bedeutet, dass φ von der relativen Position der ersten Zelle unabhängig wird. Damit gilt

$$
\lim_{L \to \infty} \left(\frac{\varphi(L,V,l,i)}{\sum_{i=0}^{2^{2(l+1)}} \varphi(L,V,l,i)} \right) = \frac{1}{2^{2(l+1)}} \; .
$$

Eine Analyse der Definition von δ zeigt, dass jeder Wert des Intervalls

$$
\left[\frac{4 \cdot 4^l + 2}{3} + 1, \frac{4 \cdot 4^{l+1} + 2}{3} \right] = [R_l + 1, R_{l+1}]
$$

2^{l+1} Mal vorkommt. Aus (A) folgt die Abschätzung

$$\begin{aligned}
\tilde{E}_v(l) &= \frac{1}{2^{2(l+1)}} \cdot \sum_{i=0}^{2^{2(l+1)}} \sum_{j=1}^{2^{l+1}} \gamma(V,l,i,j) \\
&= \frac{1}{2^{2(l+1)}} \cdot 2^{l+1} \cdot \sum_{i=R_l+1}^{R_{l+1}} \left\lfloor \frac{V + 2^{2(l+1)} - i}{2^{2(l+1)}} \right\rfloor \\
&= \frac{2^{l+1}}{2^{2(l+1)}} \cdot (V - R_l) \\
&= \frac{1}{2^{l+1}} \cdot (V - R_l) \ .
\end{aligned}$$

Eine vergleichbare Analyse für horizontale Separatoren vom Level h führt zu

$$\tilde{E}_h(l) = \frac{1}{2^{l+1}} \cdot (V - U_l) \ ,$$

wobei U_l die Distanz über den Pfad der Kurve zum direkten Nachbarn an einem horizontalen Separator vom Level l bezeichnet (vgl. Lemma 2.1, S. 10).

Jetzt sind wir in der Lage, alle inneren Kanten zweifach von der Anzahl der Kanten aller Zellen zu subtrahieren. Die Oberfläche S ergibt sich zu

$$S = 4V - 2\sum_{i=0}^{\infty} \max\{\tilde{E}_v(l), 0\} - 2\sum_{i=0}^{\infty} \max\{\tilde{E}_h(l), 0\} \ .$$

Um auf die Maximum-Funktion verzichten zu können, wird für die weitere Auswertung dieses Terms $\mathbb{N}$ in Intervalle der Klassen

$$\begin{aligned}
I_1 &= \left[\frac{2 \cdot 4^k + 1}{3}, \frac{4 \cdot 4^k + 2}{3} \right[\quad \text{und} \\
I_2 &= \left[\frac{4 \cdot 4^k + 2}{3}, \frac{2 \cdot 4^{k+1} + 1}{3} \right[
\end{aligned}$$

aufgeteilt. Die Größe der Oberfläche ist für alle Partitionen p in Intervallen der Klasse I_1

$$\begin{aligned}
S_1 &\leq 4V - \sum_{i=0}^{k} \frac{1}{2^i} \left(V - \frac{2 \cdot 4^i + 1}{3} \right) - \sum_{i=0}^{k-1} \frac{1}{2^i} \left(V - \frac{4 \cdot 4^i + 2}{3} \right) \\
&= 4V - \sum_{i=0}^{k} \left(\frac{1}{2^i} \left(V - \frac{1}{3} \right) - \frac{2}{3} \cdot 2^i \right) - \sum_{i=0}^{k-1} \left(\frac{1}{2^i} \left(V - \frac{2}{3} \right) - \frac{4}{3} \cdot 2^i \right) \\
&= 4V - \left(V - \frac{1}{3} \right) \left(2 - \frac{1}{2^k} \right) + \frac{2}{3} \left(2^{k+1} - 1 \right) \\
&\qquad - \left(V - \frac{2}{3} \right) \left(2 - \frac{1}{2^{k-1}} \right) + \frac{4}{3} \left(2^k - 1 \right) \\
&= \frac{3}{2^k} V + \frac{8}{3} 2^k - \frac{5}{3} \frac{1}{2^k}
\end{aligned}$$

und in Intervallen der Klasse I_2

$$\begin{aligned}
S_2 &\leq 4V - \sum_{i=0}^{k} \frac{1}{2^i}\left(V - \frac{2\cdot 4^i + 1}{3}\right) - \sum_{i=0}^{k} \frac{1}{2^i}\left(V - \frac{4\cdot 4^i + 2}{3}\right) \\
&= 4V - \sum_{i=0}^{k} \left(\frac{1}{2^i}\left(V - \frac{1}{3}\right) - \frac{3}{2}\cdot 2^i\right) - \sum_{i=0}^{k} \left(\frac{1}{2^i}\left(V - \frac{2}{3}\right) - \frac{4}{3}\cdot 2^i\right) \\
&= 4V - \left(V - \frac{1}{3}\right)\left(2 - \frac{1}{2^k}\right) + \frac{2}{3}\left(2^{k+1} - 1\right) \\
&\quad - \left(V - \frac{2}{3}\right)\left(2 - \frac{1}{2^k}\right) + \frac{4}{3}\left(2^{k+1} - 1\right) \\
&= \frac{2}{2^k}V + 4\cdot 2^k - \frac{1}{2^k}\ .
\end{aligned}$$

□

Die Ergebnisse des Lemma A.1 stimmen mit denen aus Lemma 3.2 (S. 42) überein. Damit ist es möglich, über beide Wege den Beweis in Satz 3.7 (S. 44) zu führen.

Literaturverzeichnis

[1] Agarwal, P. K. und J. Erickson: *Geometric Range Searching and Its Relatives*. In: *Advances in Discrete and Computational Geometry*, Bd. 223 d. Reihe *Contemporary Mathematics*, S. 1–56. American Mathematical Society, 1999.

[2] Alber, J.: *Lokalitätseigenschaften diskreter raumfüllender Kurven: Informatikrelevante Ergebnisse*. Studienarbeit, Universität Tübingen, Juli 1997.

[3] Alber, J. und R. Niedermeier: *On Multidimensional Curves with Hilbert Property*. Theory of Computing Systems, 33(4):295–312, 2000.

[4] Aluru, S. und F. E. Sevilgen: *Parallel Domain Decomposition and Load Balancing Using Space-Filling Curves*. In: *International Conference on High-Performance Computing*, S. 230–235, 1997.

[5] Anderheggen, E., D. Ekchian, K. Heiduschke und P. Bartelt: *A Contact Algorithm for Explicit Dynamic FEM-Analysis*. In: *Contact Mechanics, Computational Techniques, Proc. 1st Intl. Conf.*, S. 271–283, 1993.

[6] Asano, T., D. Ranjan, T. Roos, E. Welzl und P. Widmayer: *Space-Filling Curves and Their Use in the Design of Geometric Data Structures*. Theoretical Computer Science, 181(1):3–15, 1997.

[7] Battiti, R., A. Bertossi und A. Cappelletti: *Multilevel Reactive Tabu Search for Graph Partitioning*. Techn. Ber. UTM 554, Univ. Trento, Italy, 1999.

[8] Beckmann, N., H.-P. Kriegel, R. Schneider und B. Seeger: *The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles*. In: *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, S. 322–331, 1990.

[9] Bentley, J. L. und J. H. Friedman: *Data Structures for Range Searching*. ACM Computing Surveys, 11(4):397–409, 1979.

[10] Berger, M. und S. Bokhari: *Partitioning Strategy for Nonuniform Problems on Multiprocessors*. IEEE Transactions on Computers, 37(12):570–580, 1987.

[11] Bonet, J. und J. Peraire: *An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems*. Intl. Journal for Numerical Methods in Engineering, 31:1–17, 1991.

[12] Brown, K., S. Attaway, S. Plimpton und B. Hendrickson: *Parallel Strategies for Crash and Impact Simulations*. Computer Methods in Applied Mechanics and Engineering, 184:375–390, 2000.

[13] Bui, T. und C. Jones: *Finding Good Approximate Vertex and Edge Partitions is NP-Hard*. Information Processing Letters, 42(3):153–159, 1992.

[14] Camacho, G. und M. Ortiz: *Adaptive Lagrangian Modelling of Ballistic Penetration of Metallic Targets*. Computer Methods in Applied Mechanics and Engineering, 147:269–301, 1997.

[15] Chochia, g., M. Cole und T. Heywood: *Implementing the Hierarchical PRAM on the 2D Mesh: Analyses and Experiments*. In: *7th IEEE Symposium on Parallel and Distributed Processing*, S. 587–595. IEEE Computer Science Press, Okt. 1995.

[16] Cohen, J. D., M. C. Lin, D. Manocha und M. K. Ponamgi: *I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments*. In: *Proc. ACM Intl. 3D Graphics Conf.*, S. 189–196, 1995. http://www.cs.unc.edu/∼dm.

[17] de Berg, M., M. van Kreveld, M. Overmars und O. Schwarzkopf: *Computation Geometry: Algorithms and Applications*, Kap. 5: Orthogonal Range Searching. Springer, 2. Aufl., 2000.

[18] Diekmann, R.: *Load Balancing Strategies for Data Parallel Applications*. Doktorarbeit, Universität Paderborn, 1998.

[19] Diekmann, R., J. Hungershöfer, M. Lux, L. Taenzer und J.-M. Wierum: *Efficient Contact Search for Finite Element Analysis*. In: *Proc. European Congress on Computational Methods in Applied Sciences and Engineering*, 2000. CD-ROM.

[20] Diekmann, R., J. Hungershöfer, M. Lux, L. Taenzer und J.-M. Wierum: *Using Space Filling Curves for Efficient Contact Searching*. In: *Proc. of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, 2000. CD-ROM.

[21] Diekmann, R., D. Meyer und B. Monien: *Parallel Decomposition of Unstructured FEM-Meshes*. Concurrency: Practice and Experience, 10(1):53–72, Jan 1998.

[22] Diekmann, R., B. Monien und R. Preis: *Using Helpful Sets to Improve Graph Bisections*. In: *Interconnection Networks and Mapping and Scheduling Parallel Computations*, Bd. 21 d. Reihe *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, S. 57–73. American Society Press, 1995.

[23] Feng, Y. T. und D. R. J. Owen: *An Augmented Spatial Digital Tree Algorithm for Contact Detection in Computational Mechanics*. Intl. Journal for Numerical Methods in Engineering, 55:159–176, 2002.

[24] Fiduccia, C. M. und R. M. Mattheyses: *A Linear Time Heuristic for Improving Network Partitions*. In: *Proc. IEEE Design Automation Conf.*, S. 175–181, 1982.

[25] Fjällström, P.: *Algorithms for Graph Partitioning: A Survey*. Linkoping Electronic Articles in Computer and Information Science, 3(10), 1998.

[26] Fjällström, P., J. Petersson, L. Nilsson und Z.-H. Zhong: *Evaluation of Range Searching Methods for Contact Searching in Mechanical Engineering*. Intl. Journal of Computational Geometry & Applications, 8(1):67–83, 1998.

[27] Gilbert, J. R., G. L. Miller und S.-H. Teng: *Geometric Mesh Partitioning: Implementation and Experiments*. SIAM Journal on Scientific Computing, 19(6):2091–2110, 1998.

[28] Gotsman, C. und M. Lindenbaum: *On the Metric Properties of Discrete Space-Filling Curves*. IEEE Transactions on Image Processing, 5(5):794–797, Mai 1996.

[29] Gottschalk, S., M. C. Lin und D. Manocha: *OBBTree: A Hierarchical Structure for Rapid Interference Detection*. In: *Proc. ACM SIGGRAPH*, S. 171–180, 1996.

[30] Gregory, A., M. Lin, S. Gottschalk und R. Taylor: *H-Collide: A Framework for Fast and Accurate Collision Detection for Haptic Interaction*. In: *Proc. of IEEE Virtual Reality Conference*, S. 38–45, 1999.

[31] Griebel, M. und G. Zumbusch: *Parallel Multigrid in an Adaptive PDE Solver Based on Hashing and Space-Filling Curves*. Parallel Computing, 25:827–843, 1999.

[32] Hamada, K. und Y. Hori: *Octree-Based Approach to Real-Time Collision-Free Path Planning for Robot Manipulator*. In: *IEEE Intl. Workshop on Advanced Motion Control*, S. 705–710, 1996.

[33] Hansch, T.: *Paralleles Quicksort für große Datenmengen auf Gittern*. Diplomarbeit, Universität Karlsruhe, Juni 1996.

[34] Hayward, V.: *Fast Collision Detection Scheme by Recursive Decomposition of a Manipulator Workspace*. In: *Proc. IEEE Intl. Conf. on Robotics and Automation*, S. 1044–1049, 1986.

[35] Hendrickson, B. und K. Devine: *Dynamic Load Balancing in Computational Mechanics*. Computer Methods in Applied Mechanics and Engineering, 184:485–500, 2000.

[36] Hendrickson, B. und T. G. Kolda: *Graph Partitioning Models for Parallel Computing*. Parallel Computing, 26(12):1519–1534, 2000.

[37] Hendrickson, B. und R. Leland: *The Chaco User's Guide — Version 2.0*, 1995. http://www.cs.sandia.gov/∼bahendr/chaco.html.

[38] Hilbert, D.: *Über die stetige Abbildung einer Linie auf ein Flächenstück*. Mathematische Annalen, 38:459–460, 1891.

[39] Hungershöfer, J. und J.-M. Wierum: *On the Quality of Partitions based on Space-Filling Curves*. In: *International Conference on Computational Science (ICCS)*, LNCS 2331, S. 36–45. Springer, 2002.

[40] Hungershöfer, J., J.-M. Wierum und H.-P. Gänser: *Resource Management for Finite Element Codes on Shared Memory Systems*. In: *Proc. of Intl. Conf. on Computational Science and its Applications (ICCSA)*, LNCS 2667. Springer, 2003.

[41] Jagadish, H.: *Linear Clustering of Objects with Multiple Attributes*. In: *Proceedings of the ACM SIGMOD Conference*, S. 332–342, 1990.

[42] Jagadish, H.: *Analysis of the Hilbert Curve for Representing Two-Dimensional Space*. Information Processing Letters, 62:17–22, 1997.

[43] Kaklamanis, C. und G. Persiano: *Branch-and-Bound and Backtrack Search on Mesh-Connected Arrays of Processors*. Mathematical Systems Theory, 27(5):471–489, 1994.

[44] Karypis, G.: *Multi-Constraint Mesh Partitioning for Contact/Impact Computations*. Techn. Ber. 03-022, University of Minnesota - Computer Science and Engineering, 2003.

[45] Karypis, G. und V. Kumar: *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. SIAM Journal on Scientific Computing, 20(1):359–392, 1998.

[46] Kellermeier, T.: *Lastausgleich und Ressourcenverteilung für parallele Applikationen auf Shared-Memory-Systemen*. Diplomarbeit, Universität Paderborn, Juli 2000.

[47] Kernighan, B. und S. Lin: *An Efficient Heuristic Procedure for Partitioning Graphs*. The Bell System Technical Journal, 49(2):291–307, 1970.

[48] Klosowski, J. T., M. Held, J. S. B. Mitchell, H. Sowizral und K. Zikan: *Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs*. IEEE Transactions on Visualization and Computer Graphics, 4(1):21–36, 1998.

[49] Lennerz, C., E. Schömer und T. Warken: *A Framework for Collision Detection and Response*. In: *11th European Simulation Symposium and Exhibition*, S. 309–314. SCS Publishing, 1999.

[50] Lux, M.: *Effiziente Algorithmen zur Kontaktbehandlung in Finite-Elemente-Simulationen*. Diplomarbeit, Universität Paderborn, Juli 1999.

[51] Matoušek, J.: *Geometric Range Search*. ACM Computing Surveys, 26(4):421–461, Dez. 1994.

[52] Mehlhorn, K.: *Data Structures and Algorithms, Vol. 1: Sorting and Searching*. Springer, 1984.

[53] Moon, B., H. V. Jagadish, C. Faloutsos und J. H. Saltz: *Analysis of the Clustering Properties of the Hilbert Space-Filling Curve*. IEEE Transaction on Knowledge and Data Engineering, 13(1):124–141, 2001.

[54] Netto, E.: *Beitrag zur Mannigfaltigkeitslehre*. Crelle's Journal, 86:263–268, 1879.

[55] Niedermeier, R., K. Reinhardt und P. Sanders: *Towards Optimal Locality in Mesh-Indexings*. In: *Fundamentals of Computation Theory*, LNCS 1279, S. 364–375. Springer, 1997.

[56] Niedermeier, R., K. Reinhardt und P. Sanders: *Towards Optimal Locality in Mesh-Indexings*. Techn. Ber., Universität Karlsruhe, Fakultät für Informatik, Sep. 1997.

[57] Nour-Omid, B., A. Raefsky und G. Lyzenga: *Solving Finite Element Equations on Concurrent Computers*. In: *Proc. of the Symp. on Parallel Computations and their Impact on Mechanics*, S. 209–227. The American Society of Mechanical Engineering, 1986.

[58] Oden, J. T. und A. Patra: *Problem Decomposition for Adaptive* hp *Finite Element Method*. Computing Systems in Engineering, 6(2):97–109, 1995.

[59] Oldenburg, M. und L. Nilsson: *The Position Code Algorithm for Contact Searching*. Intl. Journal for Numerical Methods in Engineering, 37:359–386, 1994.

[60] Orenstein, J.: *Spatial Query Processing in an Object-Oriented Database System*. In: *Proceedings of the ACM SIGMOD Conference*, S. 326–336, 1986.

[61] Ou, C.-W., S. Ranka und G. Fox: *Fast and Parallel Mapping Algorithms for Irregular Problems*. Journal of Supercomputing, 10:119–140, 1996.

[62] Pascucci, V. und R. J. Frank: *Global Static Indexing for Real-time Exploration of Very Large Regular Grids*. In: *Supercomputing*, 2001. CD-ROM.

[63] Peano, G.: *Sur une courbe qui remplit toute une aire plane*. Mathematische Annalen, 36:157–160, 1890.

[64] Pilkington, J. R. und S. B. Baden: *Dynamic Partitioning of Non-Uniform Structured Workloads with Spacefilling Curve*. IEEE Transaction on Parallel and Distributed Systems, 7(3):288–300, March 1996.

[65] Plimpton, S., S. Attaway, B. Hendrickson, J. Swegle, C. Vaughan und D. Gardner: *Transient Dynamics Simulations: Parallel Algorithms for Contact Detection and Smoothed Particle Hydrodynamics*. Journal of Parallel and Distributed Computing, 50:104–102, 1998.

[66] Preis, R.: *The PARTY Graphpartitioning-Library, User Manual - Version 1.99*. http://www.upb.de/fachbereich/AG/monien/RESEARCH/PART/party.html.

[67] Preis, R.: *Analyses and Design of Efficient Graph Partitioning Methods*. Doktorarbeit, Universität Paderborn, 2000.

[68] Preis, R. und R. Diekmann: *PARTY - A Software Library for Graph Partitioning*. In: *Advances in Computational Mechanics with Parallel and Distributed Processing*, S. 63–71. Civil-Comp Press, 1997.

[69] Sagan, H.: *Space-Filling Curves*. Springer, 1994.

[70] Samet, H.: *Spatial Data Structures: Quadtree, Octrees, and Other Hierarchical Methods*. Addison-Wesley, 1989.

[71] Sanders, P. und T. Hansch: *On the Efficient Implementation of Massively Parallel Quicksort*. In: *4th International Symposium on Solving Irregularly Structured Problems in Parallel*, S. 13–24. Springer, 1997.

[72] Schamberger, S. und J.-M. Wierum: *Graph Partitioning in Scientific Simulations: Multilevel Schemes versus Space-Filling Curves*. In: *Proc. of Parallel Computing Technologies (PaCT)*, LNCS 2763, S. 165–179. Springer, 2003.

[73] Schloegel, K., G. Karypis und V. Kumar: *Graph Partitioning for High Performance Scientific Simulations*. In: *The Sourcebook of Parallel Computing*. Morgan Kaufmann, 2002.

[74] Suri, S., P. M. Hubbard und J. F. Hughes: *Analyzing Bounding Boxes for Object Intersection*. ACM Transactions on Graphics, 18(3):257–277, 1999.

[75] Taylor, L. und D. Flanagan: *PRONTO 2D: A Two-Dimensional Transient Solid Dynamics Program*. Techn. Ber. SAND86-0594, Sandia Natl. Labs, 1987.

[76] Taylor, R. und P. Wriggers: *Smooth Discretization for Large Deformation Frictionless Contact*. SEMM-report, UC Berkeley, 1998.

[77] van den Bergen, G.: *Efficient Collision Detection of Complex Deformable Models using AABB Trees*. Journal of Graphics Tools, 2(4):1–13, 1997.

[78] Vanderstraeten, D., R. Neunings und C. Farhat: *Beyond Conventional Mesh Partitioning Algorithms*. In: *SIAM Conference on Parallel Processing for Scientific Computing*, S. 611–614, 1995.

[79] Walshaw, C. und M. Cross: *Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm*. SIAM Journal on Scientific and Statistical Computing, 22, 2000.

[80] Wierum, J.-M.: *Average Case Quality of Partitions Induced by the Lebesgue Indexing*. Techn. Ber. TR-002-01, Paderborn Center for Parallel Computing, http://www.upb.de/pc2/, 2001.

[81] Wierum, J.-M.: *Definition of a New Circular Space-Filling Curve – $\beta\Omega$-Indexing*. Techn. Ber. TR-001-02, Paderborn Center for Parallel Computing, http://www.upb.de/pc2/, 2002.

[82] Wierum, J.-M.: *Logarithmic Path-Length in Space-Filling Curves*. In: *Proceedings of the 14th Canadian Conference on Computational Geometry (CCCG)*, S. 22–26, 2002. http://www.cccg.ca/.

[83] Wriggers, P., L. Krstulović und J. Korelc: *Development of 2D Smooth Polynomial Frictional Contact Elements based on a Symbolic Approach*. In: *Proc. European Conference on Computational Mechanics*, München, 1999. CD-ROM.

[84] Zachmann, G.: *Virtual Reality in Assembly Simulation – Collision Detection, Simulation Algorithms, and Interaction Techniques*. Doktorarbeit, Technische Universität Darmstadt, Fachbereich Informatik, 2000.

[85] Zhong, Z. und L. Nilsson: *A Contact Searching Algorithm for General Contact Problems*. Computers and Structures, 33(1):197–209, 1989.

[86] Zhou, Y. und S. Suri: *Analysis of a Bounding Box Heuristic for Object Intersection*. In: *Proc. of 10th Symposium on Discrete Algorithms (SODA)*. ACM, 1999.

[87] Zumbusch, G.: *Load Balancing for Adaptively Refined Grids*. Techn. Ber. 722, SFB 256, University Bonn, 2001.

[88] Zumbusch, G.: *On the Quality of Space-Filling Curve Induced Partitions*. Zeitschrift für Angewandte Mathematik und Mechanik, 81, SUPP/1:25–28, 2001.