Hauptseminar Wissenschaftliches Rechnen - Tetraeder-Gitter-Verfeinerung

von Tobias Klas

Seminarleitung: Prof. Dr. Martin Rumpf, betreut durch Pascal Huber (M.Sc.)

18. Januar 2017

Rheinische Friedrich-Wilhelms-Universität Bonn

Das Ziel dieser Seminararbeit ist, einen Algorithmus zur Verfeinerung von unstrukturierten Tetraedergittern zu konstruieren, der sowohl konsistente als auch stabile Triangulierungen liefert, auch wenn diese möglicherweise nicht uniform sind. Hierzu wird zu erst die Ideen des Verfahrens an 2D-Simplices entwickelt, anschließend werden (lokale) Verfeinerungen für ein 3D-Simplex vorgestellt, d.h. Verfeinerung eines einzelnen Tetraeders. Als letztes wird daraus ein (globaler) Algorithmus abgeleitet, um ein gegebenes Tetraedergitter zu verfeinern. Der Einsatz des Verfahrens für adaptive Methoden liegt auf der Hand. Auf diesen Aspekt wird jedoch nicht tiefer eingegangen. Für den hier vorgestellten Algorithmus ist es vollkommen ausreichend zu wissen, dass eine Verfeinerung ausgeführt werden sollte, wenn der lokale Fehler eine gewisse Schranke überschreitet. Der zentrale Punkt dieser Abhandlung ist die Erarbeitung eines sinnvollen Algorithmus und dessen Eigenschaften (Konsistenz und Stabilität der erzeugten Triangulierungen, Laufzeit). Aus diesem Grund sind Vorkenntnisse aus den Bereichen Lineare Algebra und numerische Mathematik sowie der Umgang mit Algorithmen nicht nur empfehlenswert, sondern vorausgesetzt. Diese Seminararbeit basiert größtenteils auf (1). Als Ergänzungen wurden (2), (3) und (4) sowie das Skript (5) zur Vorlesung Scientific Computing 1 (2016/17) von Professor Rumpf verwendet.
Hinweise auf Tippfehler und Korrekturen bitte an s6toklas(at)uni-bonn.de.

Inhaltsverzeichnis

1. Grundlagen

In diesem Kapitel werden wir einige grundlegende Definitionen kennenlernen, die für das weitere Vorgehen fundamental sind. Sei Ω im folgenden stets ein Polygon- bzw. Polyeder-Gebiet also insbesondere Teilmenge des $\mathbb{R}^2$ bzw. $\mathbb{R}^3$.

Definition 1.1. Wir nennen eine Triangulierung

$$\mathcal{T}_k := \{T_i \mid i = 1, ..., n\},$$

wobei die T_i entweder 2D oder 3D Simplices sind (d.h. Dreiecke oder Tetraeder) *konform*, falls folgende Eigenschaften gelten:

(1) $\bigcup_{T \in \mathcal{T}_k} \overline{T} = \overline{\Omega}$.

(2) Besteht der Schnitt $T_i \cap T_j$ für $T_i, T_j \in \mathcal{T}_k$ aus nur einem Punkt, so ist dieser Punkt ein Eckpunkt von T_i und T_j.

(3) Besteht der Schnitt $T_i \cap T_j$ $T_i, T_j \in \mathcal{T}_k$ mit $i \neq j$ aus mehr als einem Punkt, so ist $T_i \cap T_j$ ein Untersimplex von T_i und T_j, d.h. eine Kante (2D,3D) bzw. eine Seitenfläche (3D).

Wir schreiben $\mathcal{T}_h$ falls jedes Element in $\mathcal{T}$ einen Durchmesser von höchstens $2h$ besitzt, dabei ist h durch

$$h := \max_{i=1,...,n} \max_{k,j=0,...,3} \left\| x_k^i - x_j^i \right\|$$

definiert. Eine Familie von Triangulierungen $\{\mathcal{T}_h\}$ heißt *uniform*, falls es eine Zahl $\kappa > 0$ gibt, so dass jedes Element T aus $\mathcal{T}_h$ einen Kreis mit Radius $\rho \geq \frac{h}{\kappa}$ enthält.

Definition 1.2 (C1)**.** Wir sagen eine Folge $(\mathcal{T}_m)_{m\in\mathbb{N}_0}$ von Triangulierungen erfüllt die *Verschachtelungseigenschaft*, falls für alle $T \in \mathcal{T}_k$ mit $k > 0$ genau ein $T' \in \mathcal{T}_{k-1}$ existiert, so dass $T \subseteq T'$ und jeder Eckpunkt von T ist entweder Eckpunkt von T' oder Mittelpunkt einer Kante von T'. Wir nennen T' *Vater* von T und T *Sohn* von T'.

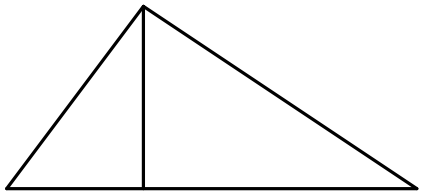


Abbildung 1.1: Verschachtelungseigenschaft verletzt

Definition 1.3 (C2)**.** Eine Folge $(\mathcal{T}_m)_{m\in\mathbb{N}_0}$ von Triangulierungen heißt *konsistent*, falls für alle k der Schnitt $T_i \cap T_j$ für alle $T_i, T_j \in \mathcal{T}_k$ entweder leer ist, eine gemeinsame Ecke, eine gemeinsame Kante oder eine gemeinsame Fläche enthält.

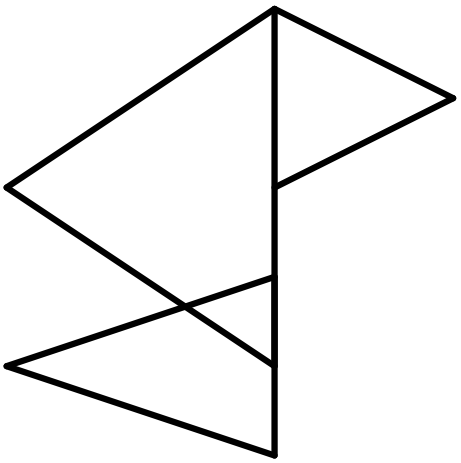


Abbildung 1.2: nicht konsistent

Definition 1.4 (C3)**.** Wir nennen eine Folge $(\mathcal{T}_m)_{m\in\mathbb{N}_0}$ von Triangulierungen *stabil*, falls für jedes k

$$\delta(T) := \frac{h(T)}{2\rho(T)}$$

für alle $T \in \mathcal{T}_k$ einheitlich beschränkt ist. Hierbei ist $h(T)$ die Länge der längsten Seite des Dreiecks T und $\rho(T)$ der Radius des Innenkreises.

Wir nehmen nun an, dass für Ω eine Initialtriangulierung $\mathcal{T}_0$ gegeben ist. Unser Ziel ist, eine Folge von Triangulierungen $(\mathcal{T}_m)_{m\in\mathbb{N}_0}$ zu konstruieren, so dass für diese Folge

die Verschachtelungseigenschaft (C1), Konsistenz (C2) und Stabilität (C3) erfüllt sind. Adaptivität, Konsistenz und Stabilität sind weitestgehend nicht kompatibel.
Unser erstes Ziel ist die Angaben einer Strategie mit der sich ein einzelner Tetraeder in acht volumengleiche Tetraeder aufteilen lässt, so dass für ein initiales Element T eine stabile und konsistente Triangulierung von T ausgehend besteht.

Definition 1.5. Eine Verfeinerungsprozedur, die stabile und konsistente Triangulierungen liefert, nennen wir *regulär*.

2. Verfeinerung von 2D-Simplices

Beispiel die Rot-Grün-Verfeinerung (RGR): Die Abbildung (2.1) zeigt die *Grün-Verfeinerung*

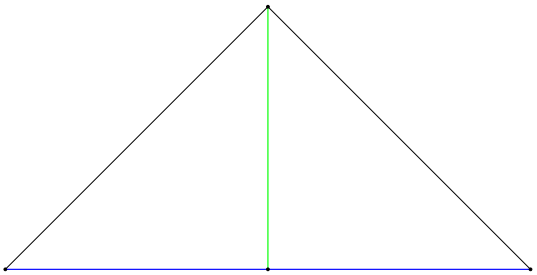


Abbildung 2.1: Grün-Verfeinerung

(GR), diese ist eine irreguläre Verfeinerung eines Dreiecks (sie ist für ein einzelnes Dreieck im Allgemeinen nicht stabil). Hierzu wird der Mittelpunkt einer Kante bestimmt und mit dem gegenüberliegenden Eckpunkt durch eine Strecke verbunden. Die blaue Kante heißt *refinement edge* (r.e.), den Eckpunkt gegenüber der r.e. nennen wir *peak*. Eine reguläre Verfeinerung ist die sogenannte *Rot-Verfeinerung*, hierbei werden die Mittelpunkte der beiden benachbarten Kanten verbunden, wie in Abbildung (2.2) illustriert. Das Problem

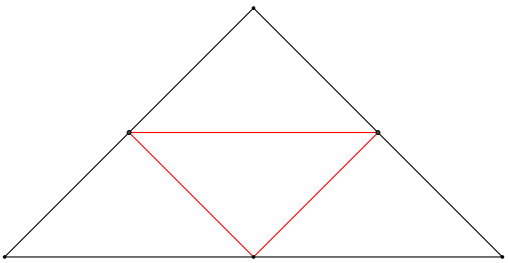


Abbildung 2.2: Rot-Verfeinerung

ist nun, wie wir im folgenden Bild sehen, dass wenn wir ein Dreieck T_1, jedoch nicht die Nachbardreiecke T_2, T_3 und/oder T_4 verfeinern, so entstehen *hängende Knoten*. Hängende Knoten sind die Mittelpunkte, die Eckpunkt eines Sohnes T_1' von T_1, die nicht Eckpunkt eines Sohns eines Nachbardreiecks sind. Der Knoten wird von dem Nachbardreieck nicht

„gestützt“, sie „hängen “gewissermaßen auf der Kante.

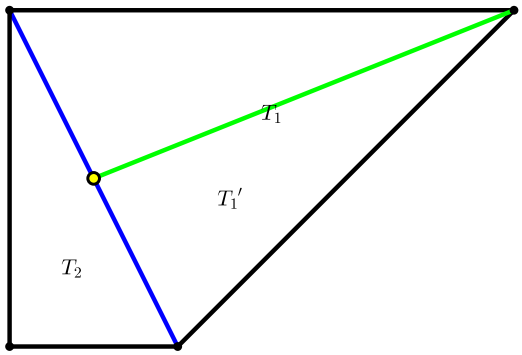


Abbildung 2.3: Hängender Knoten (gelb)

Hierfür gibt es eine Ausweg:

- Dreiecke die eine gemeinsame Verfeinerungskante haben werden beide zur selben Zeit verfeinert.
- Ein Dreieck dessen Verfeinerungskante auf dem Rand liegt wird verfeinert.

Hierzu folgendes Beispiel:

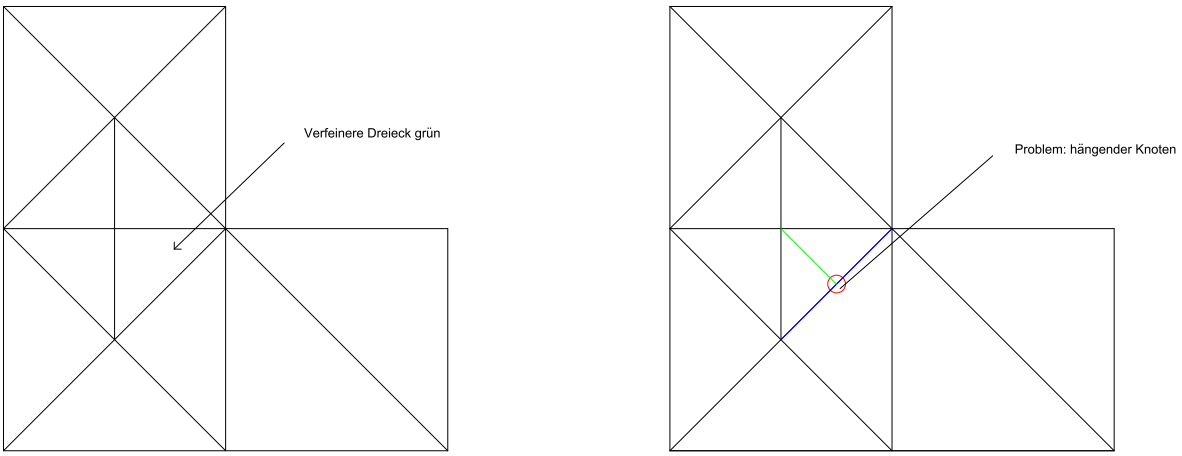


Abbildung 2.4: Verfeinerung mit hängendem Knoten

Mit obiger Überlegung erhalten wir folgende Lösung:

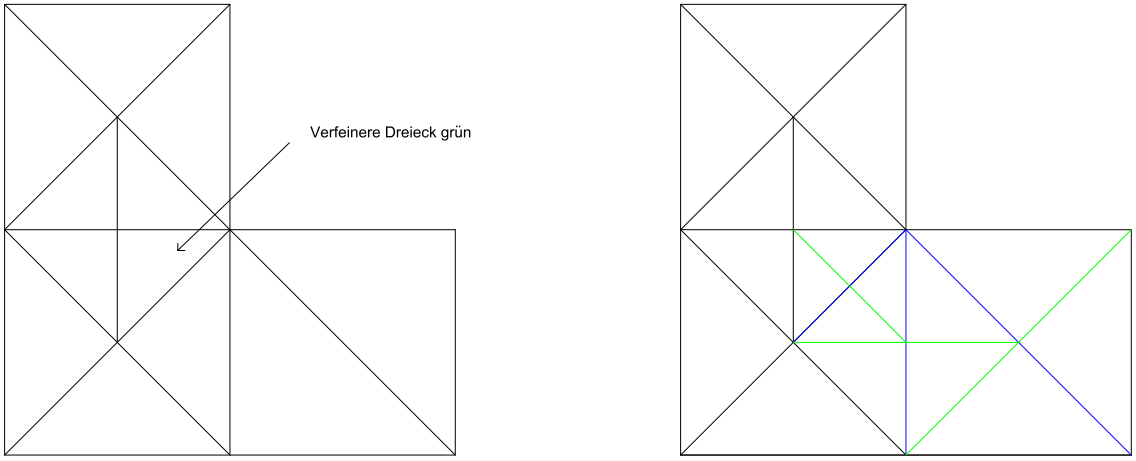


Abbildung 2.5: Verfeinerung ohne hängendem Knoten

Für eine globale Verfeinerung merken wir uns:

(C4) Irreguläre Simplices werden nie verfeinert, d.h. Simplices die aus einer irregulären Verfeinerung entstanden sind.

(C5) Ist $T \in \mathcal{T}_k \cup \mathcal{T}_{k+1}$, dann ist $T \in \mathcal{T}_l$ für alle $l \geq k$. Wird ein Simplex nicht von $\mathcal{T}_k$ nach $\mathcal{T}_{k+1}$ verfeinert, so wird es nie mehr verfeinert.

Hierbei verhindert (C4), dass irreguläre Verfeinerungen die Stabilität der regulären Verfeinerungen zerstören. (C5) sorgt dafür, dass sich die Folge von Triangulierungen eindeutig aus der Angabe der Initialtriangulierung $\mathcal{T}_0$ und einer weiteren Triangulierung $\mathcal{T}_k$ rekonstruieren lässt. Wir wollen jedoch nicht nur Verfeinerungen zulassen, sondern auch Vergröberungen. So wird zu einer Folge $\mathcal{T}_0, ..., \mathcal{T}_k$ eine Folge $\mathcal{T}_0', ..., \mathcal{T}_l'$ ausgegeben mit $\mathcal{T}_0 = \mathcal{T}_0'$ aber $l \in \{k, k+1\}$.

3. Reguläre Verfeinerung eines Tetraeders

Im folgenden sei T stets ein Tetraeder. Unser Ziel ist es nun, T in acht volumengleiche Subtetraeder $T_1, ..., T_8$ zu zerlegen. Dazu sei ein Tetraeder durch seine vier Eckpunkte identifiziert. Also

$$T = [x_0, x_1, x_2, x_3],$$

wobei $[x_0, ..., x_3]$ ein geordnetes 4-Tupel.

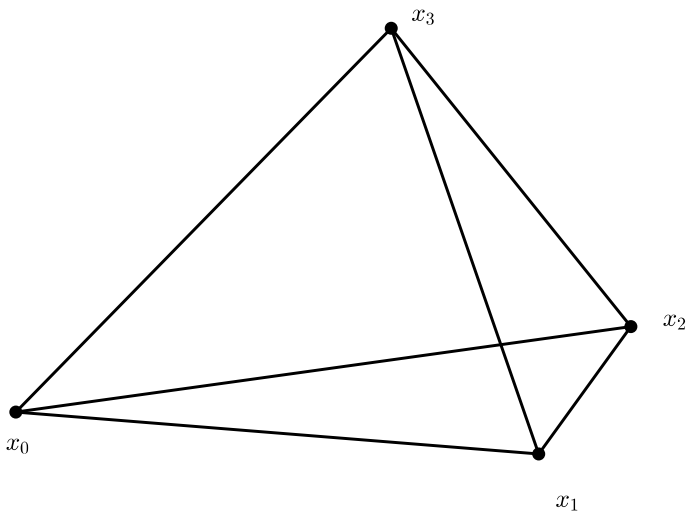


Abbildung 3.1: Der Tetraeder $T = [x_0, x_1, x_2, x_3]$

Für $0 \leq i, j \leq 3$, $i \neq j$, notieren wir den Mittelpunkt der Kante $[x_i, x_j]$ durch

$$\boxed{x_{ij} = \frac{1}{2}(x_i + x_j).}$$

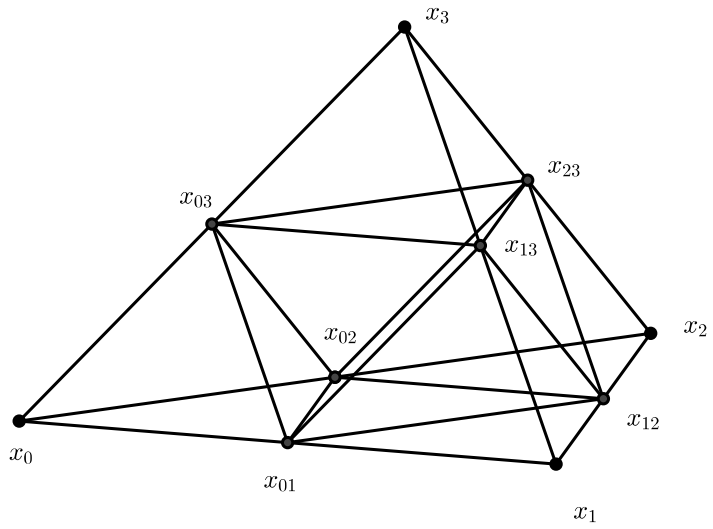


Abbildung 3.2: Aufteilung des Tetraeders $T = [x_0, x_1, x_2, x_3]$

Abbildung (3.2) setzt sich aus vier volumengleichen Tetraeder in den Ecken und einem Oktaeder zusammen. Dieser Oktaeder lässt sich wie folgt in vier volumengleiche Tetraeder zerlegen.

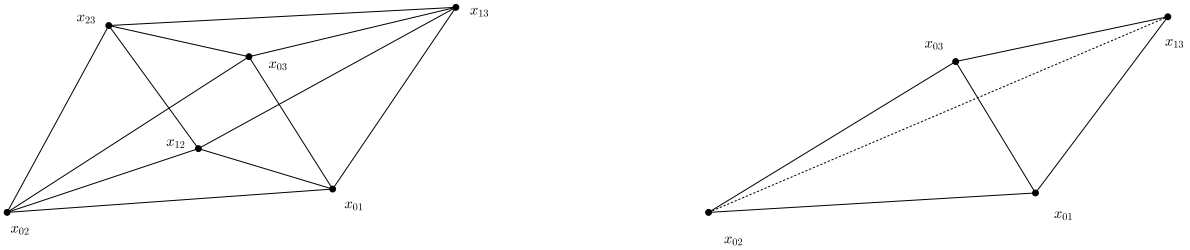


Abbildung 3.3: Innerer Oktaeder (links), rechts Tetraeder (ein Viertel des Oktaeders)

Das hier beispielhaft beschriebene Vorgehen lässt sich wie folgt in einem Algorithmus präzisieren. Diesen geben wir in Pseudocode an.

Algorithmus 3.1 (regular refinement algorithm)**.** Im Folgenden abgekürzt mit RRA.
Input: Ein Tetraeder $T = [x_0, x_1, x_2, x_3]$.
Output: Eine Zerlegung von T in acht volumengleiche Subtetraeder $T_1, ..., T_8$.

```
1  RegularRefinement(T)
2  {
3    //divide(T) into subtetrahedra T_i with 1 ≤ i ≤ 8 given by
4      T_1 := [x_0, x_01, x_02, x_03]; T_5 := [x_01, x_02, x_03, x_13];
5      T_2 := [x_01, x_1, x_12, x_13]; T_6 := [x_01, x_02, x_12, x_13];
6      T_3 := [x_02, x_12, x_2, x_23]; T_7 := [x_02, x_03, x_13, x_23];
7      T_4 := [x_03, x_13, x_23, x_3]; T_8 := [x_03, x_12, x_13, x_23];
8  }
```

Definition 3.2. Zwei Tetraeder T_1, T_2 heißen *kongruent*, falls ein $c \neq 0$, ein Vektor $\mathbf{x} \in \mathbb{R}^3$ und eine orthogonale Matrix $Q \in \mathbb{R}^{3\times 3}$ existiert, so dass

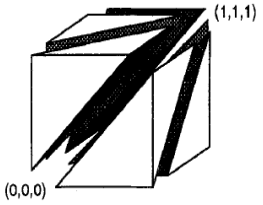
$$T_1 = \mathbf{x} + cQT_2 := \{\mathbf{x} + cQ\mathbf{x}' \mid \mathbf{x}' \in T_2\}.$$

D.h. T_1 lässt sich durch Rotationen, Translationen und Skalierungen in T_2 überführen und umgekehrt.

Offensichtlich stellt diese Kongruenz eine Äquivalenzrelation dar. Somit lassen sich alle Tetraeder in Äquivalenzklassen einteilen. Insbesondere lässt sich jedes durch RRA erzeugt Element einer von höchstens drei Äquivalenzklassen zuordnen. Dies sehen wir schon an obigen Bildern $T1, ...T4$ (die Tetraeder in den Ecken) sind alle gleich bis auf Position . Für die vier Tetraeder, die in dem Oktaeder enthalten sind, gilt, dass je zwei gegenüberliegende Tetraeder bis auf Rotation gleich sind. Also ergeben sich drei Äquivalenzklassen. Wir wollen nun RRA näher untersuchen.

Theorem 3.3 (Stabilität und Konsistenz)**.** *Für jeden (initialen) Tetraeder T liefern rekursive Ausführungen von RRA konsistente und stabile Triangulierungen von T.*

Beweis. Für den Beweis betrachten wir zuerst Triangulierungen auf dem Einheitswürfel $C = [0, 1]^3$. Mittels affiner Transformation werden wir die gewonnenen Resultate auf einen nicht-degenerierten Tetraeder übertragen.
Zunächst zerteilen wir den Einheitswürfel C in sechs Tetraeder, die sich durch Permutationen ihrer Koordinaten in einander überführen lassen.



Für alle $\pi \in \mathcal{S}_3$, ist $T_\pi = [x_\pi^0, x_\pi^1, x_\pi^2, x_\pi^3]$ definiert als die abgeschlossene konvexe Hülle von den Eckpunkten

$$\mathbf{x}_\pi^0 = (0,0,0)^T, \qquad \mathbf{x}_\pi^i = \mathbf{x}_\pi^{i-1} + e_\pi^i \text{ für i=1,2,3,} \tag{3.1}$$

wobei e^j den j-ten Standardeinheitsvektor im $\mathbb{R}^3$ darstellt. Diese Definition gibt uns folgende Darstellung an die Hand:

$$T_\pi = \left\{\mathbf{x} \in C \mid 0 \le x_{\pi(3)} \le x_{\pi(2)} \le x_{\pi(1)} \le 1, \pi \in \mathcal{S}_3.\right\} \tag{3.2}$$

Sei $\mathcal{T}_0 = \{T_\pi \mid \pi \in \mathcal{S}_3\}$ eine Triangulierung von C. Für $\pi \neq \pi'$ ist der Schnitt $T_\pi \cap T_{\pi'}$ ein gemeinsamer Subsimplex von T_π und $T_{\pi'}$ und somit $\mathcal{T}_0$ konsistent. Obiges Bild zeigt diese Zerlegung von C in sechs Tetraeder, diese Triangulierung wird *Kuhn-Triangulierung* genannt. Eine andere Triangulierung $\mathcal{T}_1$ von C definieren wir folgendermaßen:
$\mathcal{B}$ sei kanonische Zerlegung von C in acht Würfel der Kantenlänge $\frac{1}{2}$. Hierbei wollen wir jeden Würfel $C_\mathbf{x} \in \mathcal{B}$ durch seine dem Ursprung am nächsten gelegene Ecke identifizieren. Also

$$\mathcal{B} := \left\{C_\mathbf{x} \mid \mathbf{x} \in \left\{0, \frac{1}{2}\right\}^3\right\}, \tag{3.3}$$

wobei

$$C_\mathbf{x} := \mathbf{x} + \frac{1}{2}C := \left\{\mathbf{x} + \frac{1}{2}\mathbf{x}' \mid \mathbf{x}' \in C\right\} \quad \text{für } \mathbf{x} \in \left\{0, \frac{1}{2}\right\}^3. \tag{3.4}$$

Die Kuhn-Triangulierung von einem Subwürfel $C_\mathbf{x}$ ist durch die Tetraeder $T_{\mathbf{x},\pi} := \mathbf{x} + \frac{1}{2}T_\pi$, $\pi \in \mathcal{S}_3$, gegeben. Somit ist

$$\mathcal{T}_1 := \left\{T_{\mathbf{x},\pi} \mid \mathbf{x} \in \left\{0, \frac{1}{2}\right\}^3, \pi \in \mathcal{S}_3\right\} \tag{3.5}$$

eine Triangulierung von C. Aus der Konsistenz von $\mathcal{T}_0$ und dem Fakt, dass die Kuhn-Triangulierung benachbarter Subwürfel $C_\mathbf{x}, C_{\mathbf{x}'}$ eine eindeutige 2D-Triangulierung einer gemeinsamen Fläche liefert, folgt die Konsistenz von $\mathcal{T}_1$.
Wir wollen nun zeigen, dass $\mathcal{T}_1$ im Sinne von $(C1)$ eine Verfeinerung von $\mathcal{T}_0$ ist. Für ein beliebiges $\mathbf{x} \in \left\{0, \frac{1}{2}\right\}^3$ und $\pi \in \mathcal{S}_3$, suchen wir eine Permutation $\pi^* = \pi^*(\mathbf{x}, \pi)$, so dass $T_{\mathbf{x},\pi} \subseteq T_{\pi^*}$. Sei daher $0 \le k \le 3$ die Anzahl der Koordinateneinträge x_i in $\mathbf{x}$, für die $x_i = \frac{1}{2}$ gilt. Es folgt, dass es k eindeutig bestimmte Indizes $i_1, ..., i_k \in \{1,2,3\}$ gibt, sodass

$$1 \le i_1 < ... < i_k \le 3, \text{ und } x_{\pi(i_1)} = ... = x_{\pi(i_k)} = \frac{1}{2}, \tag{3.6}$$

wobei die verbleibenden $3-k$ Indizes $i_{k+1}, ..., i_3 \in \{1,2,3\}$ so angeordnet werden können, dass

$$1 \le i_{k+1} < ... < i_3 \le 3, \text{ und } x_{\pi(i_{k+1})} = ... = x_{\pi(i_3)} = 0 \tag{3.7}$$

gilt. Im folgenden wollen wir die Fälle $k = 0$ und $k = 3$ außer Acht lassen, da für sie die folgenden Ungleichungen keinen Sinn machen. Wir definieren π durch $\pi^*(j) = \pi(i_j)$, $1 \le j \le 3$. Aus den rechten Seiten von 3.6 und 3.7 erhalten wir:

$$x_{\pi^*(1)} = ... = x_{\pi^*(k)} = \frac{1}{2} \text{ und } x_{\pi^*(k+1)} = ... = x_{\pi^*(3)} = 0. \tag{3.8}$$

Weiterhin gilt für jedes $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)^T \in \frac{1}{2}T_\pi$, dass $0 \leq \xi_{\pi(3)} \leq \xi_{\pi(2)} \leq \xi_{\pi(1)} \leq \frac{1}{2}$. Benutzten wir die linken Seiten von 3.6 und 3.7, so ergibt dies

$$0 \leq \xi_{\pi^*(k)} \leq ... \leq \xi_{\pi^*(1)} \leq \frac{1}{2}, \text{ und } 0 \leq \xi_{\pi^*(3)} \leq ... \leq \xi_{\pi^*(k+1)} \leq \frac{1}{2}. \tag{3.9}$$

Mit der Monotonie von $0 \leq x_{\pi(3)} + \xi_{\pi(3)} \leq x_{\pi(2)} + \xi_{\pi(2)} \leq x_{\pi(1)} + \xi_{\pi(1)} \leq 1$ folgt aus 3.8, 3.9 und 3.2, dass $T_{\mathbf{x},\pi} \subseteq T_{\pi^*}$. Nach Konstruktion ist jeder Eckpunkt von $T_{\mathbf{x},\pi}$ ein Eckpunkt oder ein Kantenmittelpunkt von T_{π^*} und somit ist $\mathcal{T}_1$ eine Verfeinerung von $\mathcal{T}_0$ im Sinne von (C1).
Bisher haben wir die Existenz einer Verfeinerungsmethode für die Elemente der Kuhn-Triangulierung $\mathcal{T}_0$ von C gezeigt. Diese Methode erzeugt die selbe Triangulierung $\mathcal{T}_1$, die wir durch Zerlegung von C in acht Subwürfel $C_\mathbf{x} \in \mathcal{B}$ und eine anschließende weitere Kuhn-Triangulierung erhalten. Folgendes Bild zeigt diese beiden äquivalenten Wege.

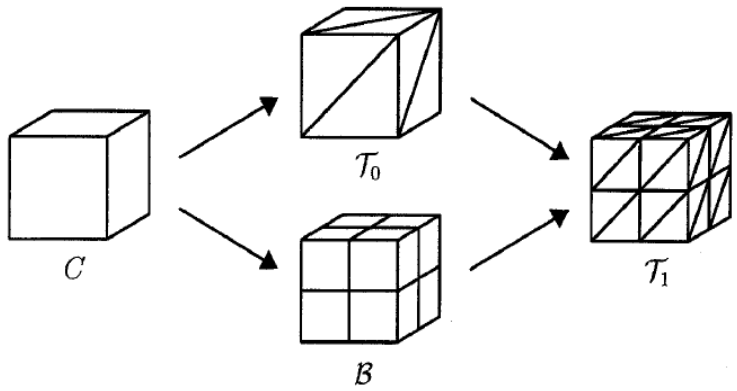


Abbildung 3.4: Zwei Wege, um einen Würfel zu triangulieren

Wir müssen nun noch zeigen, dass $\mathcal{T}_1$ genau die Triangulierung ist, die durch Anwendung von RRA auf alle Elemente $T_\pi \in \mathcal{T}_0$ erzeugt wird. Seien die Knoten wie in 3.1 nummeriert. Betrachten wir zuerst das Referenzelement $T_0 := T_{\pi(\mathrm{id})} = T_{(1,2,3)}$ mit den Eckpunkten $(0,0,0)^T, (1,0,0)^T, (1,1,0)^T$ und $(1,1,1)^T$. Benutzen wir nun RRA, um T_0 zu verfeinern. Hierbei lassen sich die Söhne $T_{0,i}$, $1 \leq i \leq 8$ durch

$$T_{0,i} = \mathbf{x}_i + \frac{1}{2}T_{\pi_i}, \qquad \mathbf{x}_i \in \left\{0, \frac{1}{2}\right\}^3, \ \pi \in \mathcal{S}_3, \ 1 \leq i \leq 8 \tag{3.10}$$

darstellen. Die Anordnung der $T_{0,i}$ entspreche der in RRA beschriebenen, dann sind die Startknoten $\mathbf{x}_i$ und Permutationen π_i, $1 \leq i \leq 8$, gegeben durch

$$\mathbf{x}_1 = (0,0,0)^T, \ \ \mathbf{x}_2 = \mathbf{x}_5 = \mathbf{x}_6 = \left(\frac{1}{2}, 0, 0\right)^T,$$

$$\mathbf{x}_3 = \mathbf{x}_7 = \mathbf{x}_8 = \left(\frac{1}{2}, \frac{1}{2}, 0\right)^T, \qquad \mathbf{x}_4 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T \tag{3.11}$$

und

$$\pi_1 = \pi_2 = \pi_3 = \pi_4 = \pi_{\mathrm{id}}$$
$$\pi_5 = (2,3,1), \quad \pi_6 = (2,1,3), \quad \pi_7 = (3,1,2), \quad \pi_8 = (1,3,2). \tag{3.12}$$

Achtung (a,b,c) ist keine Zykelschreibweise. Darstellung 3.10 impliziert $T_{0,i} \in \mathcal{T}_1$ für $1 \leq i \leq 8$. Für $i \neq j$ zeigen 3.11 und 3.12, dass entweder $\mathbf{x}_i \neq \mathbf{x}_j$ oder $\pi_i \neq \pi_j$ gilt. Somit korrespondieren $T_{0,i}$ und $T_{0,j}$ zu zwei verschiedenen Elementen $T_{\mathbf{x}_i,\pi_i}, T_{\mathbf{x}_j,\pi_j} \in \mathcal{T}_1$, welche ein paarweise disjunktes Inneres haben. Weiterhin liefert uns 3.10, dass die Volumina aller Söhne aufsummiert das Gesamtvolumen von T_0 ergeben. Somit erhalten wir aus der Konvexität von T_0, dass die erzeugte Verfeinerung von T_0 mit der durch $\mathcal{T}_1$ induzierten übereinstimmt.
Um dieses Resultat für alle Elemente in $\mathcal{T}_0$ zu erhalten, assoziieren wir jedes $\pi \in \mathcal{S}_3$ mit seiner Permutationsmatrix $P_\pi := \left(\delta_{i,\pi(j)}\right)_{i,j=1,\ldots,3}$. Somit erhalten wir $T_\pi = P_\pi T_0$ und insbesondere für die Eckpunkte

$$\mathbf{x}_\pi^j = P_\pi \mathbf{x}_{\pi_{\mathrm{id}}}^j, \qquad 0 \leq j \leq 3.$$

Die Anwendung von RRA auf T_π liefert $T_{\pi,i} = P_\pi T_{0,i}$ für $1 \leq i \leq 8$. Für $\pi_i \in \mathcal{S}_3$ ist die zu $\pi \circ \pi_i$ gehörende Permutationsmatrix gegeben durch $P_{\pi\circ\pi_i} = P_\pi P_{\pi_i}$. Somit ist durch

$$T_{\pi,i} = P_\pi T_{0,i} = P_\pi \mathbf{x}_i + \frac{1}{2} T_{\pi\circ\pi_i} \tag{3.13}$$

eine analoge Aussage zu 3.10 gegeben. Nun impliziert $P_\pi \mathbf{x}_i \in \left\{0, \frac{1}{2}\right\}^3$, dass $T_{\pi,i} \in \mathcal{T}_1$ für alle $1 \leq i \leq 8$,$\pi \in \mathcal{S}_3$. Mit vorheriger Argumentation folgt sofort, dass die erzeugte Verfeinerung von T_π mit einer durch $\mathcal{T}_1$ induzierten übereinstimmt. Wenn diese Aussage für alle $\pi \in \mathcal{S}_3$, dann ist $\mathcal{T}_1$ die durch RRA generierte Verfeinerung von $\mathcal{T}_0$. Nun müssen wir noch verifizieren, dass die Nummerierung der Knoten von $T_{0,i}$ in RRA mit der durch T_{π_i} korrespondiert,d.h. die j-te Ecke von $T_{0,i}$ ist durch $\mathbf{x}_i + \frac{1}{2}\mathbf{x}_{\pi_i}^j$gegeben. Diese Eigenschaft bleibt unter Permutation invariant und somit ist diese gültig für jedes Element in $\mathcal{T}_1$. Wird nun RRA rekursiv auf die Elemente $\mathcal{T}_1$ angewendet, so folgt induktiv, dass die erzeugten Trinagulierungen $\mathcal{T}_k$, $k \geq 0$, von C durch

$$\mathcal{T}_k = \left\{ T_{\mathbf{x},\pi} = \mathbf{x} + 2^{-k} T_\pi \mid \mathbf{x} \in \left\{0, 1\cdot 2^{-k}, \ldots, (2^k - 1)\cdot 2^{-k}\right\}^3, \quad \pi \in \mathcal{S}_3 \right\} \tag{3.14}$$

gegeben ist. Das gleiche Resultat erhalten wir, wenn wir zuerst C in 8^k Subwürfel mit Kantenlänge 2^{-k}aufteilen, welche dann mit der Kuhn-Triangulierung verfeinert werden.
Um den Beweis abzuschließen müssen wir die Resultat für den Einheitswürfel auf einen nicht-degenerierten Tetraeder übertragen. Sei nun $T = [\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3]$ ein nicht-degenerierter Tetraeder und sei $F : T_0 \to T$ eine affine Transformation, die T_0 bijektiv auf T abbildet und insbesondere $\mathbf{x}_{\pi_{\mathrm{id}}}^j \mapsto \mathbf{x}^j$ für $0 \leq j \leq 3$. F bildet alle Eckpunkte und Kantenmittelpunkte von T_0 auf die Eckpunkte und Kantenmittelpunkte von T ab. Daher liefert rekursive Anwendung von RRA auf T Triangulierungen

$$\mathcal{T}_k(T) = \left\{ F(\hat{T}) \mid \hat{T} \in \mathcal{T}_k(C), \hat{T} \subseteq T_0 \right\}, \quad k \geq 0. \tag{3.15}$$

Die Konsistenz von $\mathcal{T}_k(C)$ impliziert die Konsistenz von $\mathcal{T}_k(T)$ für alle $k \geq 0$. Weiterhin kann jedes $\hat{T} \in \mathcal{T}_k(C)$ jedes Levels $k \geq 0$ durch

$$\hat{T} = \hat{\mathbf{x}} + 2^{-k} T_{\hat{\pi}}$$

mit geeignetem $\hat{\mathbf{x}} \in \mathbb{R}^3$, $\hat{x} \in \mathcal{S}_3$ dargestellt werden. Die Stabilität folgt nun aus der Tatsache, dass sich die Tetraeder immer in einer von höchstens drei Äquivalenzklassen befinden. □

4. Irreguläre Verfeinerung eines Tetraeders

Analog zur Grün-Verfeinerung nenne wir das hier vorgestellte Vorgehen „green closure“(GC). In 2D gibt es offensichtlich $2^3 = 8$ Muster ein einzelnes Dreieck zu verfeinern, je nachdem ob keine, eine, zwei oder alle Kanten des Dreiecks unterteilt wurden.

(1) Keine Kante wurde unterteilt, so wird keine Verfeinerung durchgeführt.

(2) Alle Kanten wurden unterteilt, dann wird Rot-Verfeinerung ausgeführt.

(3) Eine Kante wurde unterteilt, so gibt es folgende Verfeinerungen:

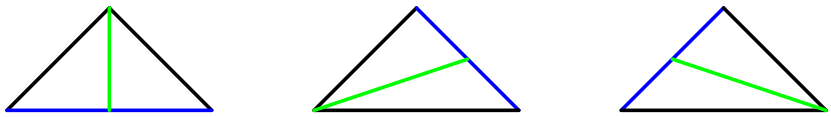


Abbildung 4.1: Drei verschiedene Grün-Verfeinerungsmuster

(4) Zwei Kanten wurden unterteilt, so gibt es folgende Verfeinerungen(manchmal Blau-Verfeinerung genannt):

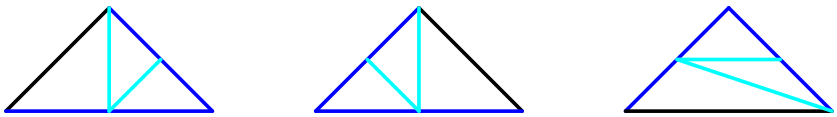


Abbildung 4.2: Drei verschiedenen Blau-Verfeinerungsmuster

Für den 3D Fall ergeben sich analog insgesamt $2^6 = 64$ Muster. Lassen wir nun das volle Muster (alle Kanten sind verfeinert wurden) und das leere Muster aus (kein Nachbarsimplex wurde verfeinert und es gibt keine verfeinerte Kante), so bleiben 62 Möglichkeiten, je nachdem, wie viele gemeinsame verfeinerte Kanten es mit einem Nachbarsimplex gibt bzw. wie ein Nachbarsimplex verfeinert wurde (welche Kanten wurden durch welches Muster verfeinert). Alle 62 Muster aufzuzählen würde den Umfang dieser Arbeit übersteigen und wäre nicht weiter zielführend. Aus Gründen der Symmetrie lassen sich diese 62 Muster in 9 Typen einteilen. Von diesen 9 Typen werden wir 4 für unsere Zwecke als zentrale Typen behandeln. Diese vier sind in folgendem Bild dargestellt.

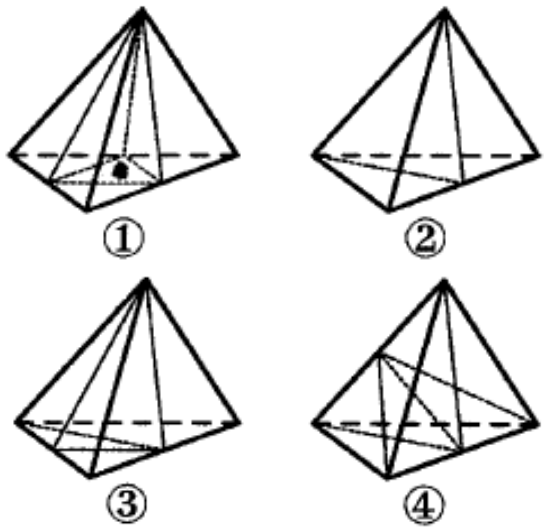


Abbildung 4.3: Vier irreguläre Verfeinerungtypen für GC in 3D

Wir verwenden

(1) Typ 1, falls ein benachbarter Simplex regulär verfeinert wurde, aber keine andere Kante unterteilt wurde,

(2) Typ 2, falls eine gemeinsame Kante refinement edge ist,

(3) Typ 3 und 4, falls zwei Kanten an einer gemeinsamen Fläche oder zwei gegenüberliegende Kanten unterteilt wurden.

Somit bleibt noch der Fall, dass drei oder mehr Kanten unterteilt wurden, die nicht zu einer gemeinsamen Seitenfläche gehören. Dies führt uns zu einer weiteren Bedingung, die wir uns für den globalen Algorithmus merken wollen.

(C6) Falls drei oder mehr Kanten von T unterteilt wurden, die nicht zu einer gemeinsamen Seitenfläche gehören, wird T regulär verfeinert.

5. Globale Verfeinerung

Wir wollen nun die lokalen Strategien so kombinieren, dass wir eine stabile Folge konsistenter Triangulierungen erhalten. Als erstes werden wir im Algorithmus verwendete sprachliche Begriffe erklären und eine abstrakte Datenstruktur einführen, die uns das Speichern aller wichtigen Informationen erlaubt. Wie diese dann konkret implementiert wird hängt natürlich von der verwendeten Programmiersprache und dem Verwendungszweck ab. Dies soll jedoch nicht Gegenstand dieser Arbeit sein.
Die zentralen Objekte in unserem Algorithmus sind Elemente (einzelne Tetraeder), Knoten (Eck- und Kantenmittelpunkte), sowie Kanten. Diese sind in folgender Weise auf einem Gitter G_k vom *Level* $k = 0, ..., k_{max}$ verteilt:

(1) G_0 enthält alle Elemente, Knoten und Kanten der Triangulierung $\mathcal{T}_0$.

(2) G_k, $k > 0$, enthält alle Elemente, Knoten und Kanten von $\mathcal{T}_k$, welche nicht schon in $\mathcal{T}_{k-1}$ in gleicher Weise enthalten sind, einschließlich der Kanten und Ecken. Zu den Objekten in G_k sagen wir, dass sie von *Level* k sind.

(3) (C5) impliziert, dass Elemente von Level k entsprechende Elemente von Level $k+1$ enthalten.

Definition 5.1. Für jeden Level-k-Knoten N gibt es mindestens ein verfeinertes Level-k-Element. Es gibt einen nachfolgenden Knoten N' mit gleichen Koordinaten aber vom Level $k+1$, dieser heißt analog zu Triangulierungen *Sohn* von N. Zwei Knoten von Level k heißen *Nachbarn*, falls sie Eckpunkte einer gemeinsamen Level-k-Kante sind. Wir sagen, dass zwei Elemente von Level k *Nachbarn* sind, falls sie eine gemeinsame Seitenfläche haben.

Im folgenden nehmen wir an, dass wir auf Knoten, Kanten, Nachbarn und Söhne der Tetraeder zugreifen können. Wir merken uns, dass die Anzahl der Kanten, die von einem Knoten ausgehen nicht a priori beschränkt ist, sondern nur aus Stabilitätsgründen eine obere Schranke nicht überschreitet.
Mit obigen Überlegungen lässt sich folgende Datenstruktur angeben:

Reference list:

- **Element** stores **Nodes** (4), **Edges** (6), Sons (≤ 8) and Neighbors (≤ 4).
- **Edge** stores Endnodes (2), Midnode (≤ 1).
- **Node** stores Edges (∞), Son (≤ 1).

Definition 5.2. Ein nicht verfeinertes Element aus G_k, $k \geq 0$, nennen wir *Blatt-Element* oder einfach *Blatt*.

Wir sagen zu Elemente T:

T is refined regularaly	T wurde durch RRA verfeinert
T is refined irregularly	T wurde irregulär verfeinert
T is unrefined	T wurde nicht verfeinert
T is marked for refinement by rule R	verfeinere T nach Regel aus Kapitel 3 oder 4
T is marked for regular refinement	T ist regulär zu verfeinern
T is marked for irregular refinement	T ist irregulär zu verfeinern
T is marked for no refinement	T ist nicht zu verfeinern
T is marked for coarsening	T ist zu vergröbern, also wird T entfernt
T is marked according to refinement	Markierung und Status von T stimmen überein

Stimmen alle Markierungen mit den Zuständen überein, so soll der Algorithmus terminieren. In gleicher Weise werden wir sagen „E is marked for refinement by T “, falls eine Kante E für eine Unterteilung markiert wurde, wenn entsprechendes Tetraeder T für Verfeinerung vorgemerkt ist. Zu einer Kante E soll es möglich sein, ohne die benachbarten Elemente zu überprüfen, ob E für eine Verfeinerung vorgemerkt ist oder nicht. Dies geschieht über einen Counter $N(E)$, der immer dann upgedatet wird, wenn sich die Markierung eines benachbarten Elements ändert.

Nun wenden wir uns dem Algorithmus zu.

Algorithmus 5.3 (global refinement algorithm)**.** Im folgenden GRA abgekürzt.
Input: Eine Folge von Gittern $G_0, ..., G_{k_{max}}$, welche wie oben beschrieben zu einer Triangulierung mit Bedingungen (C1),..., (C5) korrespondiert. Jedes Blatt-Element ist entweder „marked for regular refinement“, „marked for no refinement“oder „marked for coarsening“. Die Markierungen erfolgen gemäß eine Fehlerabschätzung. Die Markierungen aller anderen Elemente werden auf „marked according to refinement“gesetzt.
Aktion: Es wird wie folgt vorgegangen:

- Falls Blatt T „marked for regular refinement“ist, führe RRA(T) aus.
- Falls ein irreguläres Element T „marked for regular refinement“ist, ersetze es durch eine reguläres T'.
- Falls ein irreguläres Element T nicht „marked for regular refinement“ist, führe keine Aktion auf T aus.
- Regulär verfeinertes Elemente T ist „marked for no refinement“, falls alle Söhne „marked for coarsening“sind.

- Führe GC unter Bedingung (C4) bis (C6) aus.

Output: eine Folge $G'_0, ..., G'_{k'_{max}}$ mit $k'_{max} \geq 0$, $G'_0 = G_0$ und $k'_{max} \in \{k_{max}, k_{max} \pm 1\}$.

```
1   GlobalRefinement(G_0, ..., G_{k_max})
2   {
3     for k := k_max down to 0 do
4     {
5       EvaluateMarks(G_k);
6       CloseGrid(G_k);
7     }
8     for k := 0 to k_max if (G_k ≠ ∅) then do
9     {
10      if(k>0) then CloseGrid(G_k);
11      UnrefineGrid(G_k);
12      RefineGrid(G_k);
13    }
14    if(G_{k_max} = ∅) then k_max := k_max − 1;
15    else if(G_{k_max+1} ≠ ∅) then k_max := k_max + 1;
16  }
```

Wir wollen nun die in GRA verwendeten Funktionen angeben und näher erklären. EvaluateMarks berechnet und verändert die Markierungen entsprechend der oben beschriebenen Vorgehensweise (Aktion) von GRA. Damit C4 gilt werden irregulär verfeinerte Elemente auf „marked for regular refinement“, falls mindestens ein Sohn eine Kante besitzt, die „marked for refinement“ist. Dies schließt den Fall mit ein, dass dieser Sohn schon „marked for regular refinement“ist.

```
1   EvaluateMarks(G_k)
2   {
3     for all elments T ∈ G_k do
4     {
5       if(T is refined regularly and all sons of T are marked for coarsening) then
6       {
7         T is marked for no refinement;
8       }
9       if(T is refined regularly ) then
10      {
11        if(at least one edge E of a son of T is marked for refinement) then
12        {
13          T is marked for regular refinement;
14        }
15        else
16        {
17          T is marked for no refinement;
18        }
19      }
20    }
21  }
```

Ist EvaluateMarks terminiert, so sind alle Markierungen irregulärer Elemente entfernt. Falls nicht durch ein reguläres Element ersetzt, so wird jede irreguläre Verfeinerung durch

CloseGrid bestätigt.

```
1   CloseGrid(G_k)
2   {
3     let Q be the set of all regular elements T ∈ G_k having at least one edge E
4     that is marked for refinement but not marked for refinement by T;
5     while (Q ≠ ∅) do
6     {
7       choose an element T ∈ Q;
8       Q := Q \ {T}; // note Q is a set (every element belongs only once to Q)
9       CloseElement(T);
10    }
11  }
```

In der Funktion CloseGrid werden die Elemente ermittelt, die abgeschlossen werden könnten. Kandidaten dafür sind alle regulären Elemente T mit einer Kante, die „marked for refinement by $\bar{T}$“, aber nicht „by T “sind. Somit sind Elemente die „marked for regular refinement“keine Kandidaten.

```
1   CloseElement(T)
2   {
3     Search for a rule R refining exactly those edges E of T that are marked for refinement;
4     if (R is found) then
5     {
6       if (R is of type (3) and T' ∈ Q for the critical neighbor T' of T) then
7       {
8         fit R to the mark of T';
9       }
10      T is marked for refinement by R;
11    }
12    else
13    {
14      for all E of T that are not marked for refinement do
15      {
16        Q := Q ∪ {T' ≠ T | E is an edge of T'};
17      }
18      T is marked for regular refinement;
19    }
20  }
```

Für eine Element T sucht die Funktion CloseElement erst nach einer regulären oder irregulären Verfeinerungsregel R mit einer Unterteilung der Kanten von T, die genau die Kanten verwendet die „marked for refinement“sind. Existiert eine solches R, so ist T „marked for refinement by R“. Ist R von Typ 3, so muss T gesondert behandelt werden. R muss der Markierung des kritischen Nachbarn, der mit T die Kante, die „marked for refinement“ist, teilt, angepasst werden sofern der Nachbar nicht schon in Q ist und für eine Typ-3-Verfeinerung markiert ist. Falls kein geeignetes R gefunden wird, so wird (C6) angewandt und T ist „marked for regular refinement“. In diesem Fall sind alle Elemente an diesen Ecken von T, die nicht schon durch „marked for refinement“markiert wurden, neue Kandidaten, die abgeschlossenen werden können, werden zu Q hinzugefügt.
Vorsicht: CloseGrid und CloseElement verändern nur die Markierungen der Elemente,

eine Verfeinerung wurde noch nicht durchgeführt.

```
1   UnrefineGrid(G_k)
2   {
3     if (k = k_max) then G_{k+1} := ∅;
4     for all objects O ∈ G_{k+1} do
5     {
6       O is not marked for reuse;
7     }
8     for all refined elements T ∈ G_k that are marked according to refinement do
9     {
10      all sons of T, their edges and nodes are marked for reuse ;
11    }
12    for all objects O ∈ G_{k+1} that are not marked for reuse do
13    {
14      remove O from G_{k+1};
15    }
16  }
```

UnrefineGrid entfernt ungenutzte Elemente, Knoten und Kanten des folgenden Levels. Die verbleibenden Elemente sind „marked for reuse“. **Wir nehmen an, dass diese Markierungen die Verfeinerungsmarkierungen nicht beeinflussen**, denn Objekte, die „marked for reuse“sind, sind alle Söhne von Elementen, die „marked according to refinement“sind. Zu Anfang wird eine mögliches neues Gitter nächst höheren Levels initialisiert und die reuse-Markierungen für die Level-$(k+1)$-Objekte entfernt.

```
1   RefineGrid(G_k)
2   {
3     for all Elements T ∈ G_k that are marked for coarsening do
4     {
5       T is marked for no refinement;
6     }
7     for all elements T ∈ G_k that are not marked according to refinement do
8     {
9       refine T according to the rule that T is marked for;
10      for all sons T' of T do
11      {
12        T' is marked for no refinement;
13        G_{k+1} := G_{k+1} ∪ {T'};
14        find existing nodes and edges of T' in G_{k+1};
15        create missing nodes and edges of T' and add them to G_{k+1};
16        find existing neighbors of T' in G_{k+1};
17      }
18    }
19  }
```

Die Statements in RefineGrid sind weitestgehend selbst erklärend. Wenn RefineGrid beendet ist, sind alle Elemente von Level k mit „marked for according to refinement“markiert. Nach Theorem 5.5 gelten für die erzeugte neues Folge die Bedingungen (C1) bis (C5).

Theorem 5.4 (Laufzeitkomplexität)**.** *Der Arbeitsaufwand von GRA ist proportional zu der Anzahl n^*_{out} von Blättern in der erzeugten Folge $G'_0, ..., G'_{k'_{max}}$.*

Beweis. Sei n_k die Anzahl der Elemente von Level k und $n = n_0 + ... + n_{k_{max}}$ die gesamte Anzahl von Elementen der eingegebenen Folge. Die Funktionen EvaluateMarks, UnrefineGrid und RefineGrid benötigen eine limitierte Anzahl von Operationen für jedes Element. Somit hat der Aufwand dieser Funktionen in Level k die Größenordnung $\mathcal{O}(n_k)$. Selbiges gilt für CloseGrid, da jedes Element nur maximal sechsmal zu Q hinzugefügt werden kann (für jede Kante einmal). Somit liegt die Laufzeitkomplexität in $\mathcal{O}(n)$.
Sei nun n^*_{in} die Anzahl von Blättern der eingegebenen Folge. Dann impliziert das geometrische Wachstum der Anzahl an Blatt-Elementen mit zunehmender Verfeinerungstiefe, dass

$$n \leq 2n^*_{in}$$

gilt. Die lokale Verfeinerungs- und Vergröberungsregeln liefern

$$\frac{1}{8}n^*_{in} \leq n^*_{out} \leq 8n^*_{in}.$$

Dies beweist Theorem 5.4. □

Theorem 5.5 (Konsistenz und Stabilität)**.** *Sei $\mathcal{T}_0$ eine konsistente initiale Triangulierung eines Polyeder-Gebietes Ω. Dann liefert rekursive Anwendung von GRA eine Folge von Triangulierungen, die den Bedingungen (C1) bis (C5) genügt.*

Beweis. Aus der Definition der lokalen Verfeinerungsregeln folgt, dass GRA die Bedingung (C1) erfüllt. (C4) und (C5) sind nach Konstruktion klar. Mit Theorem 3.3 und (C4) folgt sofort die Stabilität von GRA. Somit bleibt noch die Konsistenz (C2) zu zeigen. Um dies zu zeigen verwenden wir Induktion über die Level k. Nach Annahme, dass $\mathcal{T}_0$ konsistent ist, folgt mit $\mathcal{T}'_0 = \mathcal{T}_0$ die Konsistenz von $\mathcal{T}'_0$. Sei nun $\mathcal{T}'_k$ konsistent für alle $0 \leq k < k'_{max}$. Nach Anwendung von CloseGrid in der ersten for-Schleife sind alle regulären Elemente in G_k für eine konsistente Verfeinerung markiert. In der zweiten for-Schleife operiert CloseGrid auf regulären Elementen, die durch RefineGrid erzeugt wurden mit einer Kante „die marked for refinement“ist. Wir zeigen nun, dass es ausreicht diese neuen Elemente zu betrachten. Sei E ein beliebige Kante von Level k, welche „marked for refinement“ist. Die Konsistenz von $\mathcal{T}'_k$ impliziert, dass E komplett von Level-k-Elementen umgeben ist. Nach Funktion EvaluateMarks sind diese regulär. Somit bleibt die Konsistenz für $\mathcal{T}'_{k+1}$ erhalten, sofern keine weitere Kante durch Anwendung von (C6) in CloseElement verfeinert wurde. Dies kann aber auch nie der Fall sein, denn nehmen wir an, dass (C6) auf ein gerade regulär erzeugtes Element T von Level k angewandt wird. Dann gibt es mindestens drei Kanten von T die „marked for refinement“sind, aber nicht zu einer gemeinsamen Seite gehören. O.B.d.A. nehmen wir an, dass (C6) das Erste mal in Level k angewendet wird. Daher existieren die drei Kanten von T und sind schon in der ersten for-Schleife „marked for refinement“. Somit hat der Vater von T schon mindestens drei verfeinerte Kanten an mehr als einer Seite. Somit ist T' aber schon ein regulär verfeinertes Element. Dies ist ein Widerspruch zur Annahme, dass T ein neues reguläres Element ist. Somit wird (C6) in der zweiten for-Schleife niemals angewandt und somit gilt Konsistenz. □

Literaturverzeichnis

[1] Jürgen Bey. *Tetrahedral Grid Refinement*. Springer-Verlag, 1995.

[2] P. Deuflhard und M. Weiser. *Adaptive Lösung partieller Differentialgleichungen*. De Gruyter Lehrbuch. De Gruyter, 2011.

[3] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer-Lehrbuch Masterclass. Springer Berlin Heidelberg, 2013.

[4] Aßfalg, Ekmekci und Kaps. Adaptive-Gitter-Verfeinerung. `https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.070/ws15_16/WiMa-Praktikum/Projekt5_RGB.pdf`, 2015.

[5] Martin Rumpf. Scientific Computing 1. `http://numod.ins.uni-bonn.de/teaching/ws16/WissRechI/script.pdf`, 2017.