Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Institut für Numerische Mathematik
Universität Ulm

Übungsblatt 7

(Besprechung Mo. 9.12.2013)

Aufgabe 16 *(Triangulierung)*

sei $F_T : \hat{T} \to T$ mit $F_T(x) = Bx+b$ die eindeutig bestimmte affine Abbildung auf das Referenzelement $\hat{T}$. Des Weiteren sei r_T der Umkreisradius und ρ_T der Inkreisradius des Dreiecks T. Für das Referenzdreieck $\hat{T}$ gilt $r_{\hat{T}} = \frac{2-\sqrt{2}}{2}$ und $\rho_{\hat{T}} = \frac{1}{\sqrt{2}}$.

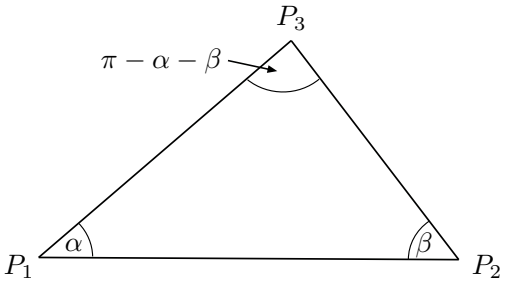
(i) Zeigen Sie die beiden Ungleichungen

$$\|B\| \leq (2+\sqrt{2})r_T \tag{1}$$

$$\|B^{-1}\| \leq \frac{1}{\sqrt{2}\rho_T}. \tag{2}$$

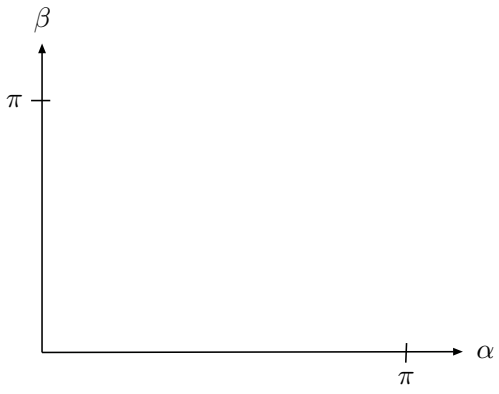
(Hinweis: Wählen Sie für die erste Ungleichung ein $\hat{x}$ mit $\|\hat{x}\| = 2\rho_{\hat{T}}$ und für die zweite Ungleichung ein x mit $\|x\| = 2\rho_T$.)

Sei nun $T := \text{conv}\{P_1, P_2, P_3\}$ ein Dreieck mit $P_1 = (0,0)^T$, $P_2 = (1,0)^T$ und Innenwinkeln α, β und $\pi - \alpha - \beta$:



(ii) Geben Sie die Matrix B der Abbildung F_T in Abhängigkeit der Winkel α und β an.

(iii) Wir wollen nun die Ungleichung (2) numerisch überprüfen, wobei wir $\alpha \leq \beta \leq \gamma := \pi - \alpha - \beta$ voraussetzen. Zeichnen Sie den zulässigen Bereich für α und β und die untenstehende Grafik ein.



Wir setzen nun die folgende Winkelbedingung voraus: $0 < \kappa_{\min} < \alpha$. Zeichnen Sie diese Bedingung in die Grafik ein.

- Schreiben Sie ein MATLAB-Skript, dass für gegebenes $\kappa_{\min}$ für alle zulässigen Kombinationen von Winkeln $\|B^{-1}\|\,\rho_T$ bestimmt. (Um die Implementierung zu erleichtern, legen sie ein Rechteck um den zulässigen Bereich und berechnen $\|B^{-1}\|\,\rho_T$ für alle Winkelkombinationen im Rechteck.)
- Plotten sie $\max \|B^{-1}\|\,\rho_T$ über $\kappa_{\min}$ für $\kappa_{\min} \in [0.01, 0.1]$. Ist obige Abschätzung (2) scharf?

Aufgabe 17 *(Adaptive Verfeinerung,* MATLAB*)*

Zur adaptiven Verfeinerung einer Triangulierung betrachten wir zwei verschiedene Verfeinerungsstrategien, die Rot-Grün-Blau-Verfeinerung und die Newest-Vertex-Bisection-Verfeinerung. Je nach Anzahl der markierten Kanten werden die Dreiecke wiefolgt verfeinert:

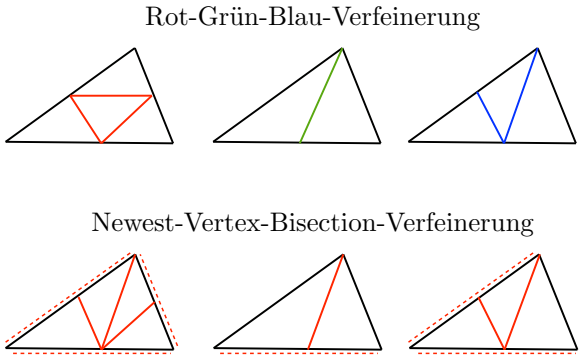
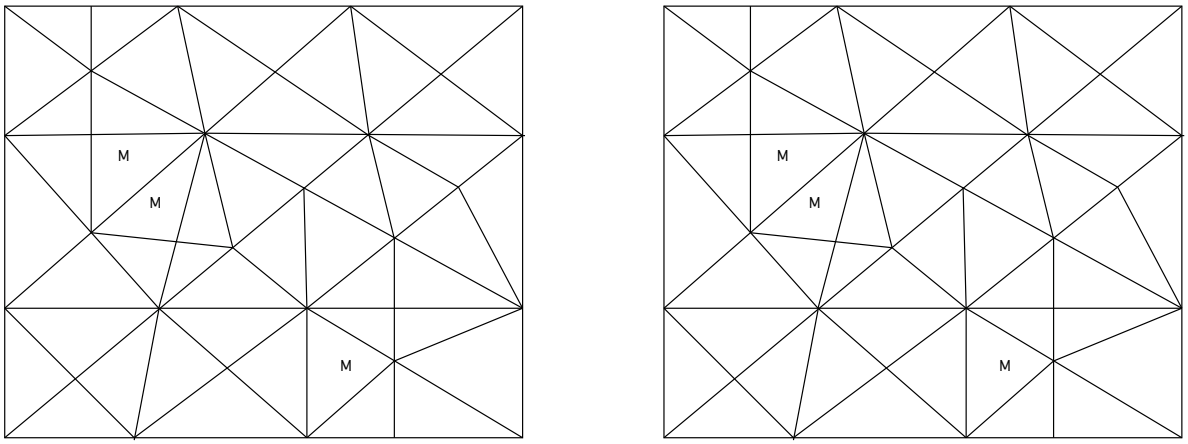


Abbildung 1: Vefeinerungsstrategien für 3 (links), 2 (mitte) und 1 (rechts) markierte Kanten.

(i) Verfeinern Sie die folgende Triangulierungen mit der RBG- und der NVB-Verfeinerung, indem Sie bei den mit **M** markierten Elementen alle drei Kanten verfeinern. Achten Sie darauf, dass Sie am Ende eine reguläre Triangluierung, also ohne hängende Knoten, erhalten.



Der Algorithmus zur adaptiven RGB-Verfeinerung ist dann wiefolgt gegeben:

- Sortiere die Kanten jedes Elements, sodass die erste Kante die Längste ist.
- Markiere die Kanten aller markierten Elemente (`markedEdges`)
- Solange noch weitere Kanten markiert werden:
  - Ist für ein Dreieck eine Kante markiert, die nicht die längste Kante ist, dann markiere auch die längste Kante.
- Verfeinere alle markierten Kanten und stelle `newElements` und `newBoundary` gemäß Abbildung 1 auf.

Der Algorithmus zur adaptiven NVB-Verfeinerung ist dann wiefolgt gegeben:

- Jedes Dreieck hat einen neuesten Knoten (bei der Ausgangs-Trianglulierung wird dieser festgelegt).

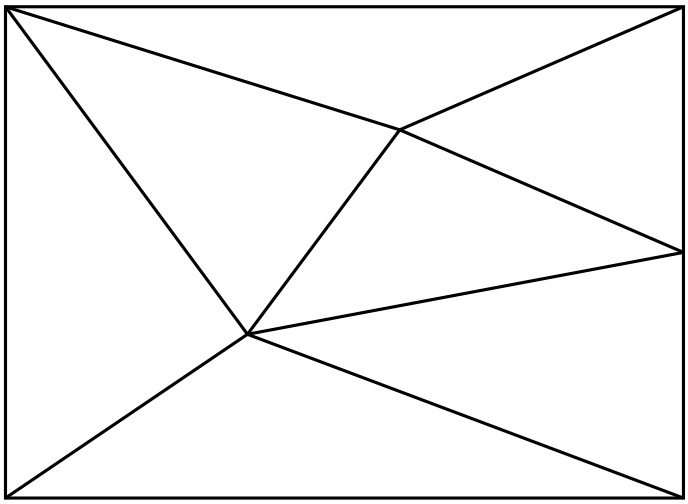
- Markiere die Kanten aller markierten Elemente (`markedEdges`)
- Solange noch weitere Kanten markiert werden:
  - Ist für ein Dreieck eine Kante markiert, die nicht dem neuesten Knoten gegenüberliegt, dann markiere auch diese Kante.
- Verfeinere alle markierten Kanten und stelle `newElements` und `newBoundary` gemäß Abbildung 1 auf.

Zur Implementierung der beiden Verfeinerungen benötigen wir neben den schon bekannten Datenstrukturen `coordinates`, `elements` und `boundary` die folgenden Hilfsstrukturen:

- `element2edges`$\in \mathbb{R}^{n_E \times 3}$: In der i-ten Zeile stehen die Indizes der Kanten des i-ten Elements (n_E bezeichnet die Anzahl der Elemente)
- `edge2nodes`$\in \mathbb{R}^{n_K \times 2}$: In der i-ten Zeile stehen die Indizes der Knoten der i-ten Kante (n_K bezeichnet die Anzahl der Kanten).
- `boundary2edges`$\in \mathbb{R}^{n_B \times 1}$: In der i-ten Zeile steht der Index der Kante der i-ten Randkante (n_B bezeichnet die Anzahl der Kanten auf dem Rand).

Diese Datenstrukturen werden von der Funktion `provideGeometricData` erzeugt.

(ii) Geben Sie zur folgenden Triangulierung eine mögliche Nummerierung der Elemente, Kanten und Knoten, sowie die oben genannten Datenstrukturen an.



(iii) Laden Sie sich die Funktion `refineRGB`, in der die Rot-Grün-Blau-Verfeinerung nach obigem Algorithmus programmiert ist, herunter und verstehen Sie den Code.

(iv) Schreiben Sie eine Funktion `refineNVB`, in der die Newest-Vertex-Bisection-Verfeinerung nach obigem Algorithmus programmiert ist. Verfahren Sie dazu wie in der Funktion `refineRBG`. Sie dürfen voraussetzen, dass der neueste Knoten des j-ten Elements immer den Index `elements(j,3)` besitzt. (Beim Verfeinern muss diese Eigenschaft erhalten bleiben!).
Hinweis: Die gegenüberliegende Kante steht dann in `element2edges(j,1)`!

Aufgabe 18 *(Bild-Approximation,* MATLAB*)*

Entgegen der sonst üblichen Vorgehensweise (das Lösen von partiellen Differentialgleichungen) wollen wir die adaptive Gitterverfeinerung dazu benutzen Kanten in Bilddaten zu erkennen. Dabei verfahren wir folgendermaßen:

- Lies eine Bilddatei pixelweise ein und konvertieren die Daten in ein Schwarz-Weiß-Bild.
- Erzeuge eine Ausgangs-Triangulierung auf der Bildfläche.
- Berechne auf jedem Dreieck die maximale Differenz zwischen den Grauwerten (`etaR = computeEtaR_pict(coorinates,elements,boundary)`).

- Markiere alle Elemente, deren Differenz in den Grauwerten größer ist als `theta` (`marked = markElementsDoerfler_pict(etaR,theta)`).
- Verfeinere alle markierten Elemente mit der RGB- oder NVB-Verfeinerung.

(i) Vervollständigen Sie die Routine `main.m` zur adaptiven Bild-Approximation.