Prof. Dr. Karsten Urban
M.Sc. Mladjan Radic, Stefan Hain
Institut für Numerische Mathematik
Universität Ulm

Numerik von PDE's
WiSe 2014/2015

Übungsblatt 10

Besprechung 14.01.2015

Aufgabe 1 (Raviart-Thomas-Element) (10 Punkte)

Es sei die folgende Menge gegeben:

$$\mathrm{RT}_h := \left\{ \tau \in L_2(\Omega)^2 \ : \ \tau|_T = \begin{pmatrix} a_t \\ b_T \end{pmatrix} + c_T \begin{pmatrix} x \\ y \end{pmatrix}, T \in \mathcal{T}_h, a_T, b_T, c_T \in \mathbb{R}, \tau \cdot v \text{ stetig an Elementgrenzen} \right\}$$

(i) $\tau \cdot v$ =const auf jeder Elementgrenze.

(ii) $\mathrm{RT}_h \subset H(\mathrm{div}; \Omega)$.
Hinweis: Satz von Green. Es genügt, wenn sie diese Aussage für eine Triangulierung bestehend aus zwei Elementen zeigen.

Aufgabe 2 (Adaptive Verfeinerung) (15 Punkte)

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:

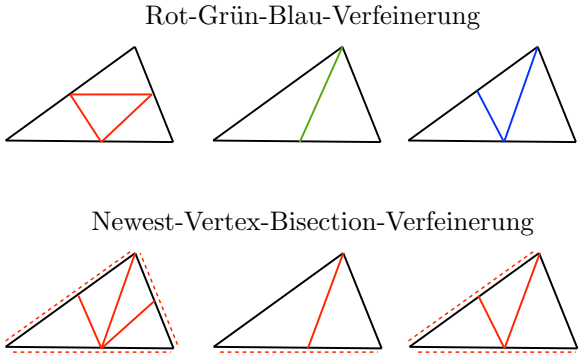
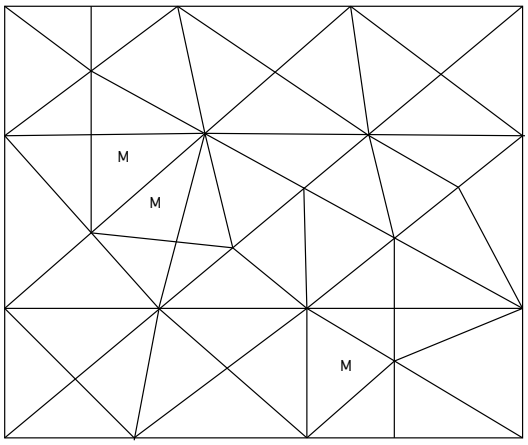
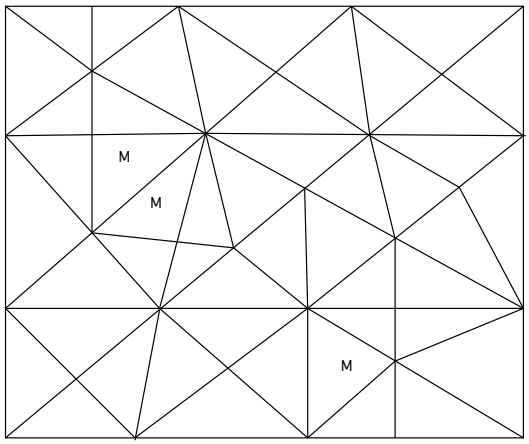


Abbildung 1: Vefeinerungsstrategien für 3 (links), 2 (mitte) und 1 (rechts) markierte Kanten.

(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:

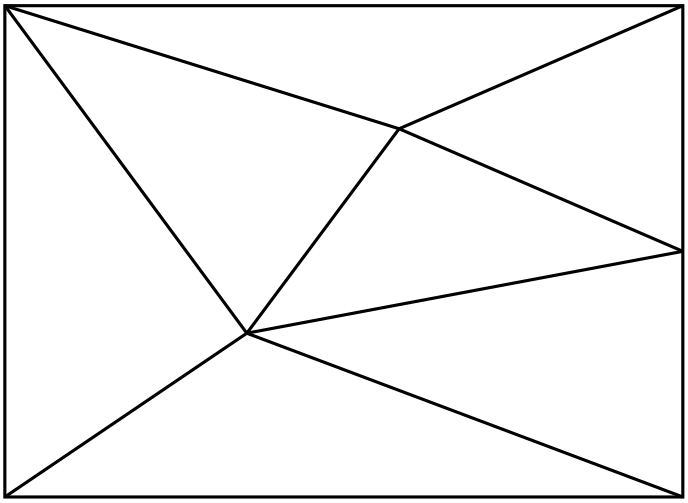
- Jedes Dreieck hat einen neuesten Knoten (bei der Ausgangs-Trianglulierung wird dieser festgelegt).
- Markiere die Kanten aller markierten Elemente (`markedEdges`)
- Solange noch weitere Kanten markiert werden:
  - Ist für ein Dreieck eine Kante markiert, die nicht dem neuesten Knoten gegenüberliegt, 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)`!