Prof. Dr. Angela Kunoth
Samuel Leweke
Numerik partieller Differentialgleichungen II — SoSe 2020

Ausgabe: 27.05.2020

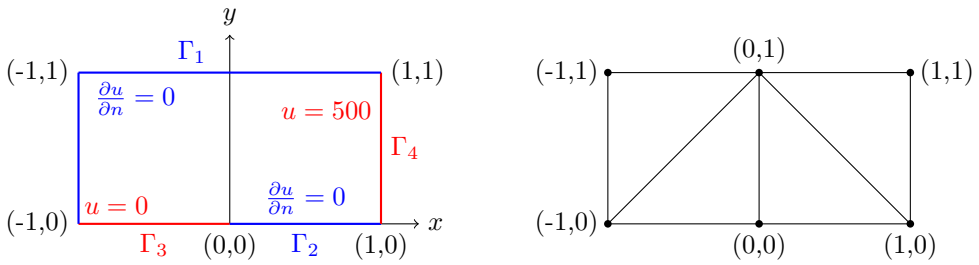
Übung 6

Abgabe: 10.06.2020, 12:00 Uhr

In dieser Übung wird das Motz-Problem [Mot47] mit linearen P1-Elementen auf einer Dreieckszerlegung adaptiv gelöst. Betrachten Sie das Problem

$$\begin{aligned} -\Delta u &= 0 && \text{in } \Omega := (-1,1)\times(0,1),\\ \frac{\partial u}{\partial n} &= 0 && \text{auf } \Gamma_1 \cup \Gamma_2 := (-1,1)\times\{1\} \cup \{0\}\times(0,1) \cup (0,1)\times\{0\},\\ u &= 0 && \text{auf } \Gamma_3 := (-1,0)\times\{0\},\\ u &= 500 && \text{auf } \Gamma_4 := \{1\}\times(0,1), \end{aligned}$$

wobei das Gebiet und die Ränder in der folgenden Abbildung beschrieben sind.

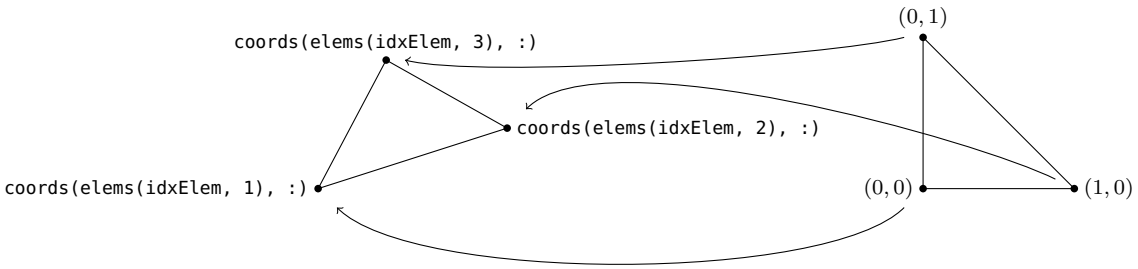


Für diese Übung wird der Matlab Code `p1afem` verwendet [FPW11]. Unter `https://www.asc.tuwien.ac.at/~praetorius/matlab/` steht sowohl der Code als auch eine Beschreibung im PDF-Format zur Verfügung. Der Code muss in dieser Übung um die Auswertung des Fehlers mit Hilfe einer Referenzlösung erweitert werden.

Aufgaben (20 Punkte)

a) Schreiben Sie eine Matlab-Funktion `[coords, elems, neumann, dirichlet] = MotzGeo()`, die die Dreieckszerlegung aus der Abbildung (rechts) erzeugt und die Ränder entsprechend markiert (in den Arrays `neumann` und `dirichlet`). Den Aufbau der zurückgegebenen Matrizen entnehmen Sie [FPW11].

b) Schreiben Sie eine Matlab-Funktion `[B, c] = MapTriangle(coords, elems, idxElem)`, welche eine affine Abbildung $\boldsymbol{x} \mapsto \mathtt{B}\boldsymbol{x} + \mathtt{c}$ berechnet, die das Referenzdreieck mit den Punkten $(0,0)$, $(1,0)$ und $(0,1)$ auf das Element mit Index `idxElem` abbildet. Zurückgegeben werden die Matrix `B` und der Vektor `c`.



c) Schreiben Sie eine Matlab-Funktion `[f, grad] = ShapeFunP1(xRef, yRef)`, welche die linearen Basisfunktionen $(x,y) \mapsto 1-x-y$, $(x,y) \mapsto \mathrm{x}$ und $(x,y) \mapsto y$ auf dem Referenzdreieck auswertet. Die Eingabe besteht aus den Spaltenvektoren `xRef` $\in \mathbb{R}^n$ und `yRef` $\in \mathbb{R}^n$, die die n Auswertungspunkte enthalten. Als

Rückgabe wird die $n \times 3$ Matrix `f` berechnet, die in jeder Zeile die Auswertung der drei Basisfunktionen an dem entsprechenden Punkt enthält. Ferner wird das $2 \times n \times 3$ Array `grad` berechnet, welches die Gradienten der Basisfunktionen an den Auswertungspunkten enthält. Die erste Dimension von `grad` enthält die Ableitungen nach x und y, die zweite Dimension gibt den Auswertungspunkt an und die letzte Dimension die Basisfunktion.

d) Schreiben Sie eine Matlab-Funktion `[L2, H1] = CalcError(coordinates, elements, uh, uRef)`, welche die Fehler `L2` $= \|u - u_h\|_{L_2(\Omega)}$ und `H1` $= |u - u_h|_{H^1(\Omega)}$ berechnet. Iterieren Sie dafür über alle Dreiecke $K \in \mathcal{T}_h$ und werten Sie die Integrale

$$\int_K (u(\boldsymbol{x}) - u_h(\boldsymbol{x}))^2 \,\mathrm{d}\boldsymbol{x}, \qquad \int_K \|\nabla u(\boldsymbol{x}) - \nabla u_h(\boldsymbol{x})\|_{\mathbb{R}^2}^2 \,\mathrm{d}\boldsymbol{x}$$

aus. Transformieren Sie dazu die Integrale auf das Referenzdreieck und verwenden Sie eine geeignete Gauß-Quadraturformel (siehe Hinweise). Die exakte Lösung und ihr Gradient werden dabei von der Funktion `uRef` ausgewertet (siehe Hinweise), die hier als Function-Handle übergeben wird. Die Koeffizienten der numerische Lösung u_h werden als Vektor `uh` übergeben.

e) Schreiben Sie ein Programm, welche das Motz-Problem sowohl mit uniformer als auch adaptiver Verfeinerung löst. Verfeinern Sie zunächst die durch `motzGeo()` erstellte Zerlegung 3 Mal uniform und nutzen Sie diese Zerlegung als Ausgang für beide Verfeinerungsarten.

Führen Sie 4 uniforme Verfeinerungen durch und berechnen Sie jeweils die Fehler $\|u - u_h\|_{L_2(\Omega)}$ und $|u - u_h|_{H^1(\Omega)}$. Nutzen Sie dazu die bereitgestellte exakte Lösung [LHL04] (siehe Hinweise).

Führen Sie 15 Iterationen des adaptiven Algorithmus mit Markierungsparameter `theta = 0.25` durch. Berechnen Sie auch hier die Fehler in den beiden Normen und plotten Sie die numerische Lösung nach der letzten Iteration. Plotten Sie außerdem die Dreieckszerlegungen zu Beginn, nach 5, 10 und 15 Iterationen (siehe Hinweise).

Stellen Sie beide Fehler für beide Verfeinerungsarten gegen die Anzahl der Freiheitsgrade in einem gemeinsamen doppelt-logarithmischen Plot dar. Schätzen Sie die Konvergenzraten und geben Sie eine entsprechende Tabelle aus.

Hinweise

- Verwenden Sie die bereitgestellte Funktion `[nodes, w] = GaussIntTri(k)`, welche die Knoten `nodes` und Gewichte `w` für die Gauß-Quadratur auf dem Referenzdreieck K_{ref} mit `k` $\in \{1, 3, 4, 6, 7, 9, 12, 13\}$ Punkten bestimmt. Damit gilt

$$\int_{K_{\mathrm{ref}}} f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} \approx \sum_{i=1}^{k} f(\mathtt{nodes}_{i,1}, \mathtt{nodes}_{i,2})\mathtt{w}_i.$$

  Der Genauigkeitsgrad ist jeweils $1, 2, 3, 4, 5, 5, 6, 7$.

- Die exakte Lösung wird von der bereitgestellten Funktion `[v,g] = MotzExact(x,y)` an den Punkten $(\mathtt{x}_i, \mathtt{y}_i)$ ausgewertet, die in den Vektoren `x`, `y` $\in \mathbb{R}^n$ übergeben werden. Die $2 \times n$ Matrix `g` enthält dabei in jeder Spalte den Gradienten ∇u ausgewertet am entsprechenden Auswertungspunkt. Der Vektor `v` enthält die Auswertung der Lösung u an den entsprechenden Punkten.

- Verwenden Sie zum Plotten der Lösung und der Dreieckszerlegung die Matlab-Funktionen `trimesh` und `trisurf`.

- Modifizieren Sie den adaptiven Algorithmus aus [FPW11] und verwenden Sie die Funktionen zur Berechnung des Fehlerindikators, zur Lösung des diskreten Problems und zum Verfeinern der Zerlegung.

- Achten Sie auf mathematisch positive Orientierung bei der Konstruktion der Geometrie. Es ist außerdem hilfreich, die Zerlegung durch Plotten (`trimesh`) zu prüfen.

Literaturverzeichnis

[FPW11] S. Funken, D. Praetorius und P. Wissgott. Efficient implementation of adaptive P1-FEM in Matlab. *Computational Methods in Applied Mathematics*, 11(4), 2011. DOI: 10.2478/cmam-2011-0026.

[LHL04] T. Lu, H. Hu und Z. Li. Highly accurate solutions of Motz's and the cracked beam problems. *Engineering Analysis with Boundary Elements*, 28(11):1387–1403, November 2004. DOI: 10.1016/j.enganabound.2004.03.005.

[Mot47] H. Motz. The treatment of singularities of partial differential equations by relaxation methods. *Quarterly of Applied Mathematics*, 4(4):371–377, Januar 1947. DOI: 10.1090/qam/18442.