Aufgabe3

November 7, 2017

1 Aufgabe 3

1.1 Ziel

Sei $\Omega = [0,1]^2$, sei $\mathcal{G}$ ein Dreiecksgitter für Ω und sei $X_\mathcal{G}$ der Raum der stückweise linearen Funktionen auf $\mathcal{G}$, d.h.

$$X_\mathcal{G} = \left\{ u \in C^0(\overline{\Omega}) \;\middle|\; u_{|T} \in \mathcal{P}_1(T) \, \forall T \in \mathcal{G} \right\},$$

wobei $\mathcal{P}_1(T)$ den Raum der linearen Polynome auf T bezeichnet.

Implementieren Sie die Lagrange-Interpolation in $I_{X_\mathcal{G}} : C^0(\overline{\Omega}) \to X_\mathcal{G}$ gegeben durch

$$I_{X_\mathcal{G}}(u)(v) = u(v) \quad \text{für alle Knoten } v \text{ von } \mathcal{G}.$$

Implementieren Sie ferner die lokale Auswertung einer Funktion $u \in X_\mathcal{G}$, d.h. gegeben ein Element $T \in \mathcal{T}$ und einen Punkt $\hat{x}$ in Referenzelement $\hat{T}$, implementieren Sie die Auswertung $u(F_T(\hat{x}))$. Visualisieren Sie Interpolation auf einem Gitter mit $4 \cdot 4 \cdot 2$ Dreiecken.

Verwenden Sie diese Auswertung, um den L^2-Fehler

$$e_h^2 := \int_\Omega |u - I_{X_\mathcal{G}}(u)|^2$$

mit Hilfe einer Quadratur der Ordnung 4 zu approximieren. Interpolieren Sie nun die Funtion u auf drei (globale) Verfeinerungen des Gitters und berechnen Sie die experimentelle Konvergenzordnung. Verwenden Sie dabei für die Gitterweite h die maximale Kantenlänge.

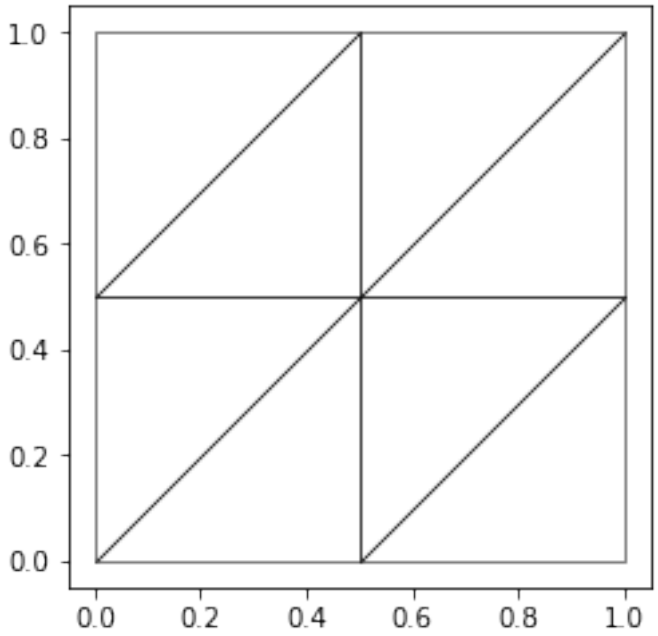
Testen Sie Ihre Implementierung, an folgenden Funtionen interpolieren:

$$\begin{aligned}
u_1(x,y) &= \sin(\pi x)\, \cos(\pi y) \\
u_2(x,y) &= \left|(x,y) - (\tfrac{1}{2}, \tfrac{1}{2})\right|^2 \\
u_3(x,y) &= \chi_{B_{\frac{3}{11}}(\frac{5}{11},\frac{5}{11})}(x,y) \left(\tfrac{3}{11} - \left|(x,y) - (\tfrac{5}{11}, \tfrac{5}{11})\right|\right)
\end{aligned}$$

1.2 Beispiel: Ein Dreiecksgitter für das Einheitsquadrat

Um ein DUNE Dreiecksgitter zu erzeugen, verwenden wir `aluSimplexGrid` und erzeugen es mit einer `cartesianDomain` (vgl. Aufgabe 2).

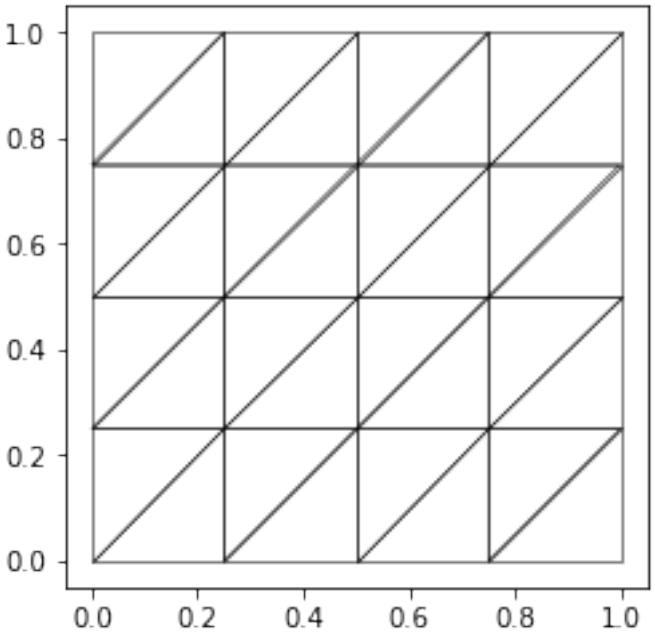
```
In [1]: from dune.grid import cartesianDomain
        from dune.alugrid import aluSimplexGrid

        grid = aluSimplexGrid(cartesianDomain([0,0], [1,1], [2,2]))
        grid.plot()
```

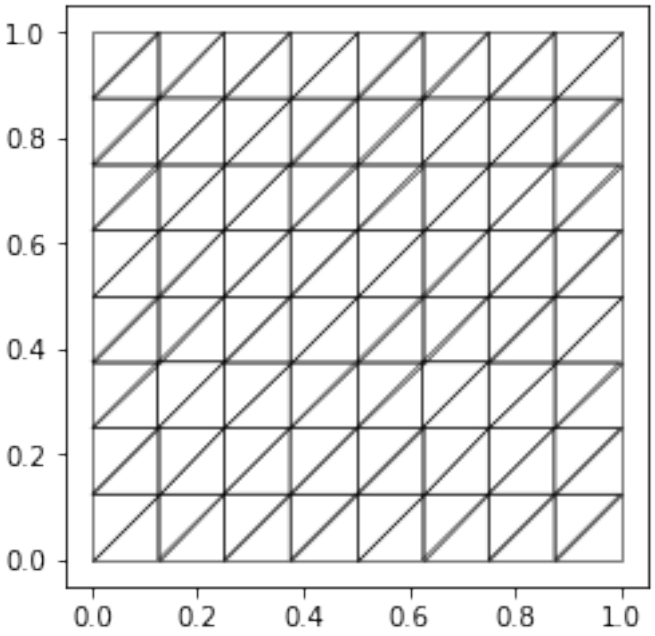


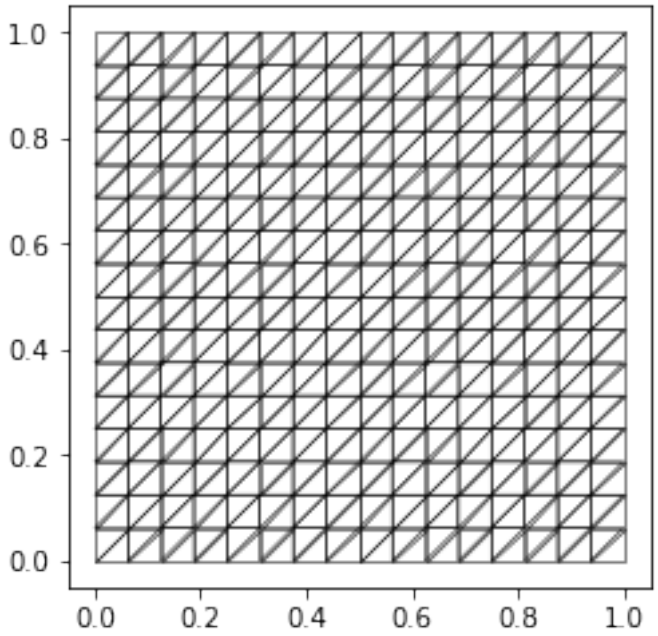
2 Beispiel: Gitterverfeinerung

DUNE Gitter sind in der Lage, sich zu global zu verfeinern, d.h. jedes Element in kleinere Elemente zu zerlegen. Die Wahl der Verfeinerungsregel liegt dabei bei der Gitterimplementierung. Im Fall von `aluSimplexGrid` wird jedes Dreieck in 4 kongruente Dreiecke zerteilt.

```
In [2]: for i in range(3):
            grid.hierarchicalGrid.globalRefine(1)
            grid.plot()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



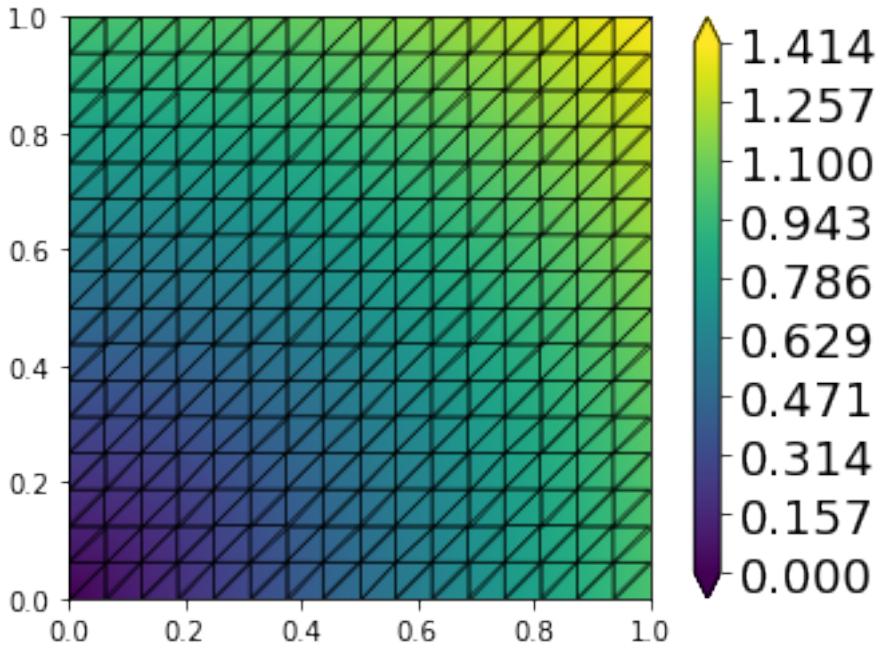
2.1 Beispiel: Eine Gitterfunktion durch lokale Auswertung

Um eine Gitterfunktion mit lokaler Auswertung zu definieren, schreiben wir zunächst eine Funktion, die die lokale Auswertung ausführt. Für die Function $u(x) = |x|$ sähe diese wie folgt aus:

```
In [3]: def u(e, x):
            return e.geometry.position(x).two_norm
```

Um eine solche Funktion in eine Gitterfunktion zu verwandeln, die wir an DUNE weitergeben können, dekorieren wir sie einfach als solche:

```
In [4]: from dune.grid import gridFunction

        @gridFunction(grid)
        def u(e, x):
            return e.geometry.position(x).two_norm

        grid.plot(u)
```



Nun ist u keine einfach Python-Funktion mehr, sondern ein DUNE Objekt.
Bemerkung: Lambda-Funktionen können nicht dekoriert werden.

3 Lösung