Aufgabe5

November 29, 2017

1 Aufgabe 5

Ziel ist die Triangulierung beliebiger Gebiete $\Omega \subset \mathbb{R}^2$. Exemplarisch werden wir ein Segment des Einheitskreises gegeben durch einen Winkel α triangulieren:

$$\Omega = \left\{ r(\cos\varphi, \sin\varphi) \;\middle|\; r \in [0,1], \varphi \in [0,\alpha] \right\}.$$

Entgegen obiger Verwendung werden wir den Winkel α in Grad angeben.
Dazu führen wir folgende Schritte durch:

1. Um die Qualität einer Triangulierung zu bewerten wollen wir den kleinsten Innenwinkel verwenden. Schreiben Sie dazu eine Funktion `minAngle`, die diesen für ein DUNE-Gitter berechnet. Testen Sie das Ergebnis anhand einer `cartesianDomain`.

2. Für den Fall $\alpha = 270°$ implementieren Sie folgende einfache Konstruktion:

   - Erzeugen Sie Gitter mit den Eckpunkten $P = \{(0,0), (1,0), (0,1), (-1,0), (0,-1)\}$.
   - Verfeinern Sie dieses Gitter 3 mal.
   - Erzeugen Sie daraus ein neues Gitter, indem Sie die Norm der Knotenpunkte derart skalieren, dass Punkte auf dem Rand der konvexen Hülle von P auf den Einheitskreis abgebildet werden.
   - Visualisieren Sie dieses Gitter und berechnen Sie den kleinsten Innenwinkel.

3. Gegeben ein α, erzeugen Sie eine geeignete Menge von Punkten und verwenden Sie `scipy.spatial.Delaunay`, um aus diesen Punkten eine Delaunay-Triangulierung zu erzeugen. Entfernen Sie ggf. Elemente, die nicht zu Ω gehören. Versuchen Sie, den kleinsten Innenwinkel möglichst groSS zu bekommen; er sollte mindestens $30°$ betragen.

4. Tragen Sie in einer Grafik den Winkel α gegen den kleinsten Innenwinkel Ihrer Delaunay-Triangulierung auf.

1.1 Beispiel: Einfache Triangulierung

Als Beispiel wollen wir das Segment $[0°, 225°]$ des Einheitskreises durch n Dreiecke triangulieren, die den Ursprung mit einem Liniensegment auf den Rand des Einheitskreises verbinden.

```
In [1]: alpha = 225
        n = 45
```

Dazu erzeugen wir eine Liste von von Punkten

$$x_i = (\cos i\omega, \sin i\omega),$$

Die Dreiecke sind dann gegeben durch

$$T_i = conv\{0, x_i, x_{i+1}\}.$$

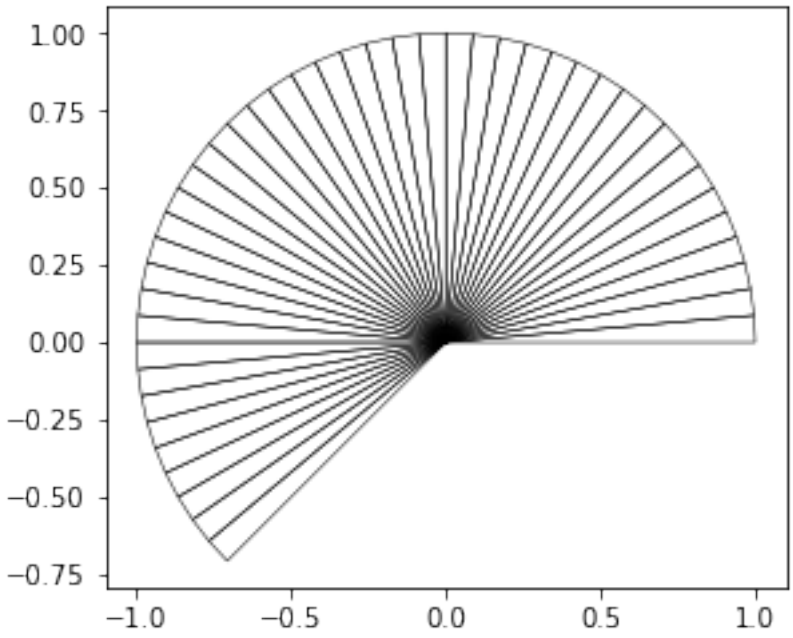
In der Praxis codieren wir die Dreicke durch das Tupel seiner Eckpunktindizes. Dazu müssen wir die Liste der Punkte um den Ursprung ergänzen und die Eckpunkte dann passend indizieren:

```
In [2]: from math import cos, sin, pi

        omega = (alpha*pi) / (180*n)
        points = [[0, 0]] + [[cos(omega*i), sin(omega*i)] for i in range(n+1)]
        triangles = [(0, i+1, i+2) for i in range(0, n)]
```

Gegenen diese zwei Listen, können wir ein DUNE-Gitter wie folgt erzeugen:

```
In [3]: from dune.alugrid import aluSimplexGrid
        grid = aluSimplexGrid({'vertices': points, 'simplices': triangles})
        grid.plot()
```

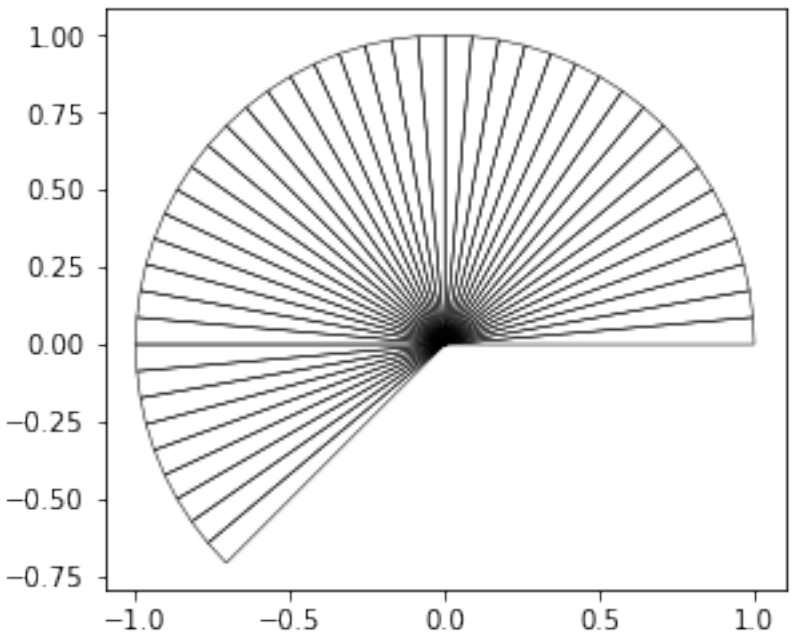


Dies eine schlechte Triangulierung von Ω, da der kleines Innenwinkel etwa 5° beträgt.

1.2 Beispiel: Ein DUNE-Gitter aus einem DUNE-Gitter erzeugen

Um Gitter zu erzeugen verwenden wir häufig andere Gitter als Basis. Die dazu notwendigen Listen aus Punkten und Dreiecken können wir aus einem DUNE-Gitter wie folgt extrahieren:

```
In [4]: indexSet = grid.indexSet
        points = [None] * indexSet.size(2)
        for v in grid.vertices:
            points[indexSet.index(v)] = v.geometry.center
        triangles = [grid.indexSet.subIndices(e, 2) for e in grid.elements]
        grid = aluSimplexGrid({'vertices': points, 'simplices': triangles})
        grid.plot()
```

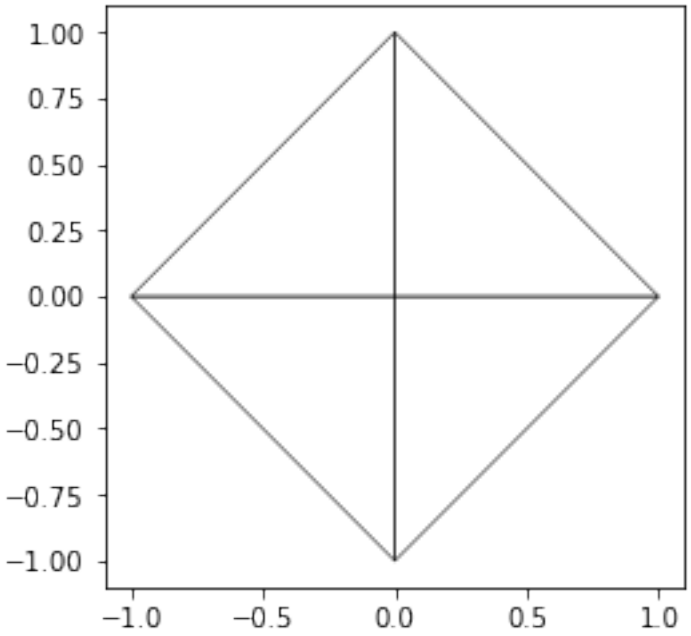


1.3 Beispiel: Delaunay-Gitter

Gegeben eine Menge von Punkten kann man eine sog. Delaunay-Triangulierung ihrer konvexen Hülle mittels `scipy.spatial.Delaunay` berechnen. Dazu gehen wir etwa wie folgt vor:

```
In [5]: import numpy
        from scipy.spatial import Delaunay

        points = [(0, 0), (1, 0), (0, 1), (-1, 0), (0, -1)]
        points = numpy.asarray(points)
        triangles = Delaunay(points).simplices
        grid = aluSimplexGrid({'vertices': points, 'simplices': triangles})
        grid.plot()
```



Um jetzt etwa das Kreissegment $[0°, 270°]$ zu erreichen, müssen wir das untere, rechte Dreieck weglassen. Diese Triangulierung lässt sich aber einfacher von Hand erzeugen.

Für weitere Details verweisen wir auf die Dokumentation von `scipy` und `numpy`.

2 Lösung