# Einfuehrung

October 20, 2017

## 1 Einführung

### 1.1 Organisatorisches

Es wird jede Woche eine Übungsaufgabe in Form eines Jupyter-Notebooks ausgegeben. Diese kann von der Vorlesungsweibseite **https://aam.uni-freiburg.de/agkr/lehre/ws17/pde0/** heruntergeladen werden.

Die Übungsaufgaben sollen direkt im Jupyter-Notebook bearteitet werden. Zur Abgabe schicken Sie das bearbeitete Notebook per eMail an **nolte@mathematik.uni-freiburg.de**. Bitte führen Sie alle Zellen des Notebooks vor Abgabe der Reihe nach aus.

Die Studienleistung gilt als erbracht, wenn 50% der Punkte aus den Übungsaufgaben erreicht wurden.

### 1.2 Inhalt

Ziel ist die Lösung partieller Differentialgleichungen der Form

$$-\nabla \cdot \big(D\,\nabla u\big) + m\,u = f \quad \text{in } \Omega$$

für ein Gebiet $\Omega \subset \mathbb{R}^2$ und - einen Diffusionstensor $D : \Omega \to \mathbb{R}^{2\times 2}$, - eine Wachstumsrate $m : \Omega \to \mathbb{R}$, - passenden Randbedingungen.

Dazu vollen wir die Finite-Elemente-Methode (FEM) verwenden. Dazu benötigen wir folgende Zutaten: - ein Dreiecksgitter $\mathcal{G}$ um das Gebiet $\Omega$ zu diskretisieren, - einen diskreten Funktionenraum $X_\mathcal{G}$, in dem approximative Lösung $u_\mathcal{G}$ gesucht wird, - einen Löser für (dünnbesetzte) lineare Gleichungssysteme.

### 1.3 Software

Wir konzentrieren uns auf die FEM-Diskretisierung und werden Gitter $\mathcal{G}$ und lineare Löser aus der Open-Source Software-Bibliothen DUNE (https://dune-project.org) verwenden.

Im CIP-Pool ist die DUNE-Umbegung vorinstalliert. Der Befehl

```
dune-python
```

startet Jupyter in einem Browser. Ein entsprechender Eintrag im Startmenü existiert ebenfalls.

### 1.4 Verwendung eigener Computer

Um die DUNE-Python Umgebung auf einem privaten Computer zu verwenden, installieren Sie Docker (https://www.docker.com). Der Jupyter-Server wird gestartet mit

```
docker run -d --rm -v dune:/dune -p 127.0.0.1:8888:8888 \
    registry.dune-project.org/staging/dune-python
```

und kann dann in einem Browser verwendet werden (http://localhost:8888). Das Passwort für den Jupyter-Server lautet `dune`. Den Server beenden Sie mit

```
docker stop <Containername>
```

Die Daten bleiben im Volume `dune` persistent auf Ihrem Computer.

## 2 Crash-Kurs Jupyter

### 2.1 Zellen

Jupyter-Notebooks sind eine lineare Liste von "Zellen", die Quellcode enthalten. Es gibt zwei wichtige Zelltypen: - Code-Zellen enthalten ausführbaren Code in Python 3. - Text-Zellen enthalten Dokumentation in Markdown

Nachdem wir eine Zelle editiert haben, können wir sie mit `Ctrl + Enter` ausführen.

### 2.2 Markdown-Zellen

Alle bisherigen Abschnitte sind in Wirklichkeit Text-Zellen. Wir werden diese verwenden, um die Aufgabenstellung zu erklären und den Code zu dokumentieren.

Markdown ist eine sehr einfache, intuitive Sprache zur Beschreibung von formatiertem Text. Die Möglichkeit zur Darstellung von LaTeX-Formeln macht sie für unsere Zwecke sehr geeignet.

Wir eine Text-Zelle ausgeführt, so wird der formatierte Text dargestellt.

### 2.3 Code-Zellen

In Code-Zellen schreiben wir den eigentlichen Python-Code. Wird eine solche Zelle ausgeführt, wird der Python-Code interpretiert und ausgeführt. Beginnen wir mit dem klassischen "Hello, World!"-Beispiel:

```
In [1]: print("Hello, World!")

Hello, World!
```

Nach Ausführung wird die Ausgabe unter der Zelle eingeblendet. Hat der letzte Ausdruck nicht den Typ `None`, so wird der Wert zusätzlich ausgegeben:

```
In [2]: a = 14
        b = 28
        a + b
```

```
Out[2]: 42
```

**Achtung:** Obwohl sich Zellen in beliebiger Reihenfolge ausführen lassen, verändern sie den Zustand des Python-Interpreters, indem sie etwa Variable belegen. Erst nachdem die obige Zelle ausgeführt wurde, können wir folgenden Code ausführen:

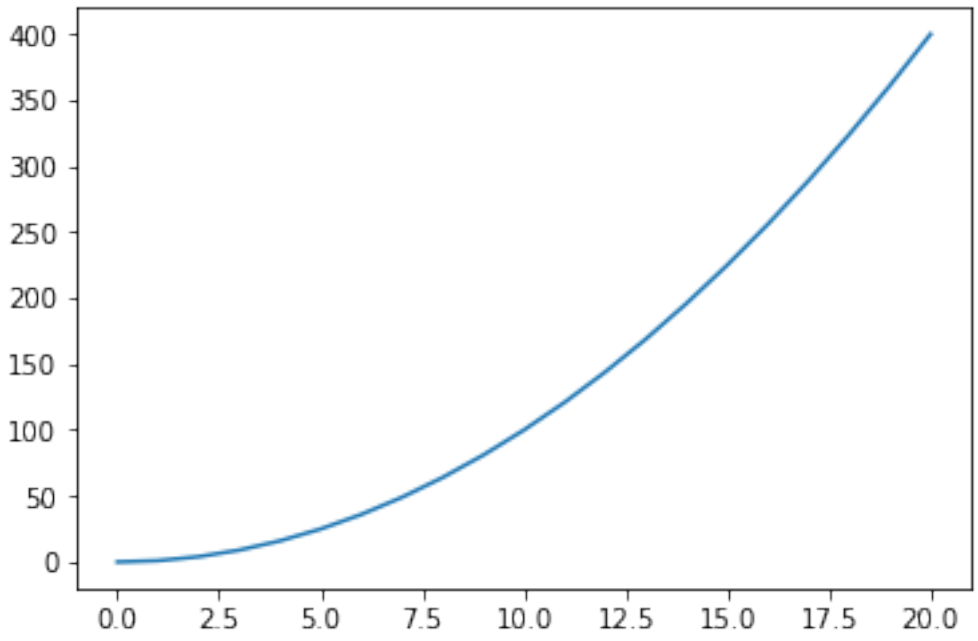
```
In [3]: print(b, "-", a, "=", b-a)

28 - 14 = 14
```

## 2.4 Visualisierung

Die Darstellung der ergebnisse ist ein wichtiger Teil der numerischen Simulation. Python bietet hierfür die `matplotlib`, die gute Unterstützung in Jupyter besitzt. Ein Beispiel:

```
In [4]: from matplotlib import pyplot
        xs = list(range(21))
        ys = [x*x for x in xs]
        pyplot.figure()
        pyplot.plot(xs, ys)
        pyplot.show()
```




Dokumentation und Beispiele finden sich auf der Webseite https://matplotlib.org.

Bei der Visualisierung von Gittern und diskreten Funktionen mit Hilfe der `matplotlib` können wir auf Hilfsfunktionen aus DUNE zurückgreifen.

# 3 Erste Schritte in DUNE

## 3.1 Ein Dreiecksgitter anlegen

Legen wir zunächst ein Dreiecksgitter für das Einheitsquadrat $[0,1]^2$ an. Zunächst beschreiben wir die Zerlegung des Gebietes:

```
In [5]: from dune.grid import cartesianDomain
        domain = cartesianDomain([0,0], [1,1], [3,3])
```

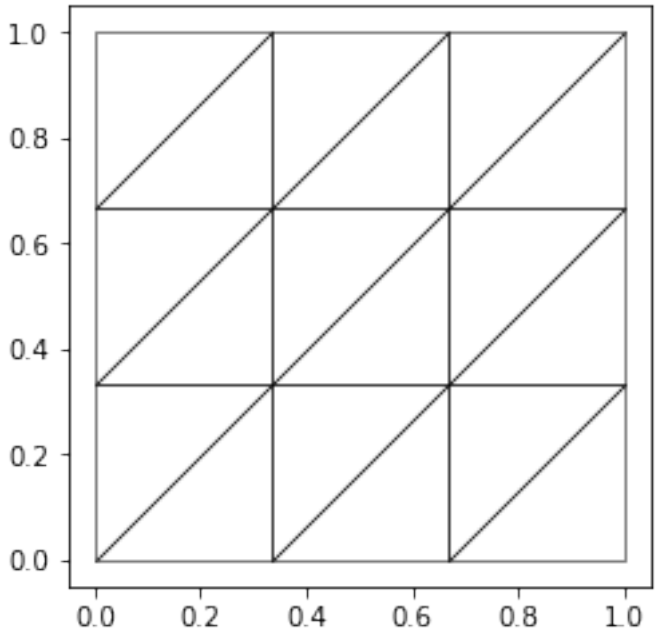
Nun erzeugen wir daraus ein Gitter:

```
In [6]: from dune.alugrid import aluConformGrid
        grid = aluConformGrid(domain)
```

**Bemerkung:** DUNE stellt verschiedene Gitterimplementierungen bereit, die als C++ Templates realisiert sind und vor der ersten Verwendung übersetzt werden müssen. Das erste Anlegen einer solchen Implementierung benötigt daher viel Zeit. Das Ergebnis der Übersetzung wird für die weitere Verwendung in einem Cache gespeichert.

Um das Gitter zu visualisieren haben wir folgende Hilfsfunktion:

```
In [7]: from dune.plotting import plotGrid
        plotGrid(grid)
```




Dieses Gitter erlaubt nun alle Operationen, die wir für eine Finite-Elemente-Methode benötigen. Ein einfaches Beispiel:

```
In [8]: print("Das Gitter besteht aus {} Dreiecken und {} Knoten.".format(grid.size(0), grid.siz
```

```
Das Gitter besteht aus 18 Dreiecken und 16 Knoten.
```

Die Funktion `size` liefert die Anzahl der "Entitäten" einer Codimension. Dabei steht 0 für Elemente (hier Dreiecke) und die Gitterdimension (hier 2) für Knoten.

Die wesentlichen Teile der Gitterschnittstelle werden wir ausführlich erarbeiten.

## 4 Hands On!

Wenden wir uns nun der ersten Aufgabe zu. Laden Sie sich das Jupyter-Notebook für Aufgabe 1 von der Vorlesungswebseite herunter:

https://aam.uni-freiburg.de/agkr/lehre/ws17/pde0/praktikum/Aufgabe1.ipynb

Legen Sie es in Ihrem Home-Verzeichnis ab und starten Sie dann `dune-python`.

*Hinweis*: Wenn Sie Docker verwenden, um einen Jupyter-Server zu starten, müssen Sie das Notebook zunächst auf den Jupyter-Server hochladen.