Computergrundlagen 2025

Blatt 11: Python und Vektorgrafiken

- Abgabetermin für die Lösungen: **11.01.2026, 20 Uhr/ für Montagsgruppe: 09.01.2026, 12 Uhr**
- Bei Fragen wendet euch bitte an eure/n Tutor/in:
  - Mo 11:30: Stephan Haag: `st170833@stud.uni-stuttgart.de`
  - Di 09:45: Julian Hoßbach: `julian.hossbach@icp.uni-stuttgart.de`
  - Mi 14:00: Julian Peters: `julian.peters@icp.uni-stuttgart.de`
  - Do 09:45: Rebecca Stephan: `rebecca.stephan@icp.uni-stuttgart.de`
  - Fr 09:45: Jonas Höpker: `st182335@stud.uni-stuttgart.de`
- Die Übungsaufgaben sollen in der Regel in **Zweiergruppen** bearbeitet werden. Nur in **begründeten Ausnahmefällen** sind Dreiergruppen möglich.
- Die Abgabe der Übungsblätter erfolgt über Ilias.
- Mit Abgabe der Lösungen erklärt Ihr, dass Ihr die Lösung euren Mitstudierenden im Rahmen der Übungsbesprechung vorstellen könnt. Um dies zu überprüfen, muss mindestens zweimal von jedem Teilnehmenden vorgetragen werden. Wenn Ihr das nicht könnt, werden euch die Punkte für die entsprechenden Aufgaben wieder abgezogen.
- **Befehle, die nicht in der Vorlesung besprochen wurden, müssen gegebenenfalls recherchiert werden.**
- **Alle erstellen Skripte (`.py` and `.ipynb`) sowie ein mit markown oder Latex erstellter Report (`.pdf`) sind Teil der Abgabe**

  Tipp: Für Abgaben in Python kann der Editor https://jupyter.org/ hilfreich sein. Hier könnt ihr euren Code *interactive* ausführen sowie mit Markdown Zellen ausführlich kommentieren. Die `*.ipynb*` Dateien können ebenfalls als Lösungen abgegeben werden und sind leicht zu validieren.

Erstellen einer Vektorgrafik (3 Punkte)

Erstelle mit Inkscape oder einem anderen Vektorgraphikprogramm deiner Wahl eine Skizze eines Physikexperiments deiner Wahl. Dokumentiere dabei wichtige Schritte mit Screenshots des Vektorgraphikprogramms.

Binde die Skizze in ein Latex Dokument ein, in dem du kurz beschreibst, was zu sehen ist.

Lineare Algebra mit NumPy: Fibonacci-Folge (4 Punkte)

Die Fibonacci-Folge ist eine Folge von natürlichen Zahlen a_n, welche durch die folgende Rekursionsrelation definiert ist:

$$a_n = a_{n-1} + a_{n-2}. \tag{1}$$

Hierbei sind die Anfangsbedingungen gegeben durch $a_1 = a_2 = 1$. Mithilfe von Vektoren und Matrizen lässt sich die Rekursionsrelation in der kompakten Form

$$\begin{pmatrix} a_{n+2} \\ a_{n+1} \end{pmatrix} = M \begin{pmatrix} a_{n+1} \\ a_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{n+1} \\ a_n \end{pmatrix} \tag{2}$$

schreiben. In dieser Aufgabe wollen wir numerisch untersuchen, wie sich das Verhältnis a_n/a_{n-1} für große n verhält.

- Vervollständige den ersten Teil des bereitgestellten Skriptes `fibonacci.py`: definiere dazu einen Vektor `v`, der mit der oben genannten Anfangsbedingung initialisiert wird sowie eine Matrix `M`. In der `for`-Schleife soll in jeder Iteration die obige Matrix-Vektor-Multiplikation durchgeführt werden und das Verhältnis a_n/a_{n-1} an die Liste `ratios` angehängt werden.
- Plotte anschließend die Folge der Verhältnisse a_n/a_{n-1} gegen n und beschreibe deine Beobachtung. Welcher Grenzwert wird erwartet?

Mithilfe eines Tricks lässt sich die Fibonacci-Folge auch auf eine andere Art und Weise untersuchen. Dazu verwenden wir die Tatsache, dass M eine symmetrische Matrix und daher diagonalisierbar ist. Man kann M also schreiben als

$$M = O^{-1}DO = O^{-1}\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}O, \tag{3}$$

wobei λ_i die Eigenwerte von M sind und die orthogonale Matrix O die Eigenvektoren von M als Spalten enthält. Die n-te Potenz von M kann damit dargestellt werden als

$$M^n = \left(O^{-1}DO\right)^n = O^{-1}D^nO = O^{-1}\begin{pmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{pmatrix}O. \tag{4}$$

- Berechne die Eigenwerte von M numerisch mithilfe von NumPy und gib sie an.

Aus der obigen Teilaufgabe sollte sich ergeben haben, dass nur einer der beiden Eigenwerte einen Betrag $|\lambda_i| > 1$ hat. Sei nun OBdA $|\lambda_2| < 1$. Damit ergibt sich

$$M^n = O^{-1}\begin{pmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{pmatrix}O \approx O^{-1}\begin{pmatrix}\lambda_1^n & 0\\ 0 & 0\end{pmatrix}O, \tag{5}$$

wobei die Approximation umso besser wird, je größer n ist.

- Implementiere eine Berechnung der Fibonacci-Folge, die auf der obigen Approximation beruht. Erstelle wieder eine Liste mit den Verhältnissen und vergleiche mit dem exakten Ergebnis von oben.

Quicksort-Algorithmus (3 Punkte)

Implementiere eine Funktion `quick_sort`, die eine Liste mithilfe des Quicksort-Algorithmus sortiert und die sortierte Liste zurückgibt.