

Algorithmik kontinuierlicher Systeme Aufgabenblatt 1 — Spaß mit Zahlen

Allgemeines:

- Die Abgabe der Programmieraufgaben erfolgt über das Exercise Submission Tool:
<https://est.cs.fau.de>
- Sie können während der Bearbeitungszeit Ihre Abgaben im EST beliebig oft aktualisieren. Nur die aktuellste Abgabe, die in der Bearbeitungszeit hochgeladen wurde, wird gewertet.
- Auf der Übungswebsite finden Sie zu jedem Übungsblatt eine Vorlage, sowie zu jeder Aufgabe eine Datei `*_test.py` mit der Sie ihre Lösungen jederzeit selbst überprüfen können.
- Damit Sie auf eine Teilaufgabe Punkte bekommen, muss sie mit Python 3.5 im CIP Pool funktionieren. Das bestehen der mitgelieferten Tests ist notwendig, aber nicht hinreichend dafür, dass Sie Punkte bekommen.

In diesem Arbeitsblatt lernen Sie die unterschiedlichen Arten von Zahlen kennen, mit denen ein Computer arbeiten kann. Sie werden feststellen, dass selbst Operationen wie $+$, $-$, \times und $/$ je nach verwendetem Zahlentyp ein völlig unterschiedliches Verhalten aufweisen. Wenn Sie diese Aufgaben gründlich machen, ersparen Sie sich später viel Verwirrung und Frustration.

Aufgabe 1 — Natürliche Zahlen (7 Punkte)

`integers.py`

- a) **Primzahlen** Schreiben Sie eine Funktion `is_prime`, die entscheidet, ob eine gegebene Zahl eine Primzahl ist.
- b) **Zahlendarstellung** Schreiben Sie eine Funktion `int2str`, die eine gegebene natürliche Zahl in eine Zeichenkette umwandelt. Das zweite Argument beschreibt dabei die Basis, in der die Zahl dargestellt werden soll.
- c) **Mirpzahlen** Schreiben Sie eine Funktion `is_emirp`, die entscheidet, ob eine gegebene Zahl eine Mirpzahl ist. Eine Mirpzahl ist eine Primzahl, deren Spiegelbild (in Dezimaldarstellung) eine **andere** Primzahl ist, z.B. die Zahlen 13 und 31. Sie können dazu natürlich die Funktionen der beiden vorherigen Aufgaben verwenden.

Aufgabe 2 — Gleitkommazahlen (4 Punkte)

`float.py`

In dieser Aufgabe werden Sie mit Gleitkommazahlen arbeiten. Im Vergleich zu ganzen Zahlen werden Gleitkommazahlen durch eine feste Anzahl Bits repräsentiert. Python verwendet standardmäßig Gleitkommazahlen vom Typ *double* gemäß dem IEEE 754 Standard. Computer können sehr effizient mit solchen Zahlen arbeiten. Der Preis dafür ist, dass bei jeder Rechenoperation Genauigkeit verloren geht.

a) Exponentialfunktion Die Exponentialfunktion e^x erscheint an vielen Stellen in der Mathematik. Eine Möglichkeit e^x auszurechnen ist über die Reihendarstellung

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \frac{1}{1} + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{6} + \dots \quad (1)$$

In der Funktion `myexp` fehlt noch **genau eine** Zeile, damit sie die Exponentialfunktion gemäß der obigen Formel approximiert. Ergänzen Sie die fehlende Zeile.

b) Numerische Differentiation Die Ableitung einer Funktion f an einer Stelle x ist definiert durch

$$f'(x) := \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}. \quad (2)$$

Im Allgemeinen kann es sehr schwierig sein, diesen Grenzwert auszurechnen. Alternativ kann man die Ableitung auch approximieren, indem man in Formel 2 ein fixes, aber sehr kleines h einsetzt, d.h.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (3)$$

Schreiben Sie eine Funktion `diff1`, die eine Funktion f , einen Wert x und den Parameter h übergeben bekommt, und mit dieser Methode die Ableitung einer Funktion abschätzt.

Eine genauere Methode zur numerischen Abschätzung der Ableitung ist es, Formel 3 sowohl mit einem Parameter h , als auch mit dem Parameter $-h$ auszuwerten, und daraus den Mittelwert zu bilden. Der resultierende Ausdruck lässt sich wie folgt vereinfachen:

$$f'(x) \approx \frac{\frac{f(x+h)-f(x)}{h} + \frac{f(x-h)-f(x)}{-h}}{2} = \frac{f(x+h) - f(x-h)}{2h} \quad (4)$$

Schreiben Sie nun auch eine Funktion `diff2`, die die Ableitung mithilfe von Formel 4 berechnet.

Aufgabe 3 — Matrizen (4 Punkte)

matrices.py

Viele bedeutsame mathematische Transformationen auf Vektorräumen sind linear. Ein linearer Operator lässt sich stets eindeutig durch eine Matrix beschreiben. In dieser Aufgabe werden Sie elementare Matrixoperationen mit dem Python-Paket NumPy durchführen. NumPy ist für wissenschaftliches Rechnen in Python unerlässlich und Sie werden es in den nächsten Übungsblättern noch oft verwenden. In den folgenden Aufgaben werden Matrizen durch Objekte vom Typ `numpy.array` beschrieben.

a) Rotationsmatrix Schreiben Sie eine Funktion `rotation_matrix`, die zu einem gegebenen Winkel ω eine 2×2 Rotationsmatrix R zurückgibt, so dass $R \cdot x$ einen zweidimensionalen Vektor x um ω gegen den Uhrzeigersinn dreht.

b) Skalierungsmatrix Schreiben Sie eine Funktion `stretch_matrix`, die zu einem gegebenen Streckungsfaktor s und einer Dimension d eine $d \times d$ Streckungsmatrix S zurückgibt, so dass für alle d -dimensionalen Vektoren x gilt $Sx = sx$.

c) Komposition Zu guter Letzt schreiben Sie eine Funktion `compose`, die beliebig viele Matrizen A_1, \dots, A_n mit den Dimensionen $d_1 \times d_2, d_2 \times d_3, \dots, d_n \times d_{n+1}$ übergeben bekommt und diese zu einer einzelnen Matrix B mit der Dimension $d_1 \times d_{n+1}$ zusammenfügt, so dass für alle d_{n+1} -dimensionalen Vektoren x gilt $A_1 \cdot A_2 \cdot \dots \cdot A_n x = Bx$.