Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M. Sc. Tino Lück
M. Sc. Joshua Doll

26. November 2025

# Einstieg in die Informatik und Algorithmische Mathematik

## Aufgabenblatt 6

Bearbeitungszeitraum: 08.12.2025 – 19.12.2025

### Aufgabe 1 (Pflichtaufgabe) *Vektoriteration*

In einem Land haben alle Familien genau einen Sohn und eine Tochter, und jeder heiratet genau einmal. Ferner lässt sich die Bevölkerung in Generationen aufteilen. In der $n$-ten Generation gibt es $x_n$ Funktionäre, $y_n$ Arbeiter und $z_n$ Bauern. Ehefrauen gehören dem Stand ihres Mannes an. Funktionärssöhne lehnen den Beruf ihres Vaters strikt ab und werden Arbeiter, Söhne von Arbeitern werden Bauern, während Bauernsöhne zu je einem Drittel den genannten drei Bevölkerungsgruppen angehören. Ist dieses System über viele Generationen hinweg stabil? Die mathematische Formulierung dieses einfachen Modells aus der Populationstheorie lautet wie folgt:

$$u^{[n+1]} := \begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1/3 \\ 1 & 0 & 1/3 \\ 0 & 1 & 1/3 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} =: Au^{[n]}, \quad n = 0, 1, 2, \ldots .$$

Der Vektor $u^{[n]}$ repräsentiert hierbei die $n$-te Generation mit den Bevölkerungsgruppen $x_n$, $y_n$ und $z_n$. Die Matrix $A$ stellt die Umstrukturierung in der Gesellschaft dar. Die $n$-te Generation ergibt sich demnach durch n-maliges Multiplizieren der Matrix $A$ mit sich selbst und dem Produkt dieser Matrix mit dem Startvektor gemäß

$$u^{[n]} = A^n u^{[0]}.$$

Veranschaulichen Sie sich zunächst das Zustandekommen der Matrix $A$. Beachten Sie, dass die Summe der Elemente in einer Spalte immer gleich eins ist. Erstellen Sie eine Klasse `Vektoriteration` , mit der Sie die Bevölkerungsentwicklung beurteilen können:

- Erstellen Sie zunächst eine Methode `readVec` mit einem eindimensionalen Feld vom Gleitkommatyp als Rückgabewert und der Dimension $DIM$ als Übergabeparameter, die den Startvektor $u^{[0]}$ einliest. Erstellen Sie ebenfalls eine Funktion `readMat`, die die Matrix $A$ einliest. Die Rückgabe soll ein zweidimensionales Feld vom Gleitkommatyp sein. Der Übergabeparameter ist ebenfalls die ganzzahlige Dimension $DIM$.

- Schreiben Sie eine Methode `writeVec` ohne Rückgabewert, die als Argument ein Feld vom Gleitkommatyp erhält. Diese Methode soll die Elemente des Vektors auf der Konsole ausgeben.

- Erstellen Sie eine Methode `matPot`, die die $n$-te Potenz $A^n$ der Matrix $A$ bestimmt. Als Argumente sollen die Matrix $A$, die ganzzahlige Dimension $DIM$ und die ganze Zahl $n$ übergeben werden. Die Methode soll ein zweidimensionales Feld vom Gleitkommatyp zurückgeben. Zur Berechnung wird eine Hilfsmatrix $Temp$ und die Ergebnismatrix $An$ benötigt, in die mithilfe einer tiefen Kopie die Matrix $A$ abgespeichert wird. Beide Matrizen sind zweidimensionale Felder vom Gleitkommatyp. Anschließend sollen mittels einer for-Schleife, die $n$ mal durchlaufen wird, die Matrizen $Temp$ und $A$ miteinander multipliziert werden und in $An$ abgespeichert werden. $An$ soll am Anfang jedes Schleifendurchlaufs $0$ gesetzt werden und am Ende des Durchlaufs soll $Temp$ gleich $An$ gesetzt werden. Am Ende wird $An$ zurückgegeben.

  **Hinweis:** Für die Multiplikation von zweidimensionalen Matrizen müssen auch zwei for-Schleifen durchlaufen werden.

- Die Funktion `matVecMult` soll das Ergebnis $b$ der Multiplikation der Matrix $A$ mit dem Vektor $u$, d.h. $Au = b$ berechnen. Dafür sollen die Matrix $A$ und der Vektor $u$ und die ganzzahlige Dimension $DIM$ als Argumente übergeben werden. Die Funktion soll das eindimensionale Feld vom Gleitkommatyp $b$ zurückgeben.

- Schreiben Sie das Hauptprogramm. Definieren Sie eine ganzzahlige Variable $DIM$, die den Wert $3$ haben soll. Erzeugen Sie die benötigten zweidimensionalen Felder vom Gleitkommatyp $A$ und $An$ und eindimensionalen Felder vom Gleitkommatyp $u0$ und $u$. Zudem soll einer ganzzahligen Variable $n$, die die Iterationszahl abspeichert, zunächst der Wert $1$ zugewiesen werden. Lesen Sie die Matrix $A$ im Hauptprogramm nur einmal ein. Erstellen Sie eine `while`-Schleife, in der Sie das Einlesen des Startvektors $u0$ und von $n$ solange wiederholen bis die Eingabe vom Benutzer mit $n \leq 0$ abgebrochen wird. In der Schleife soll mithilfe des eingelesenen Startvektors das Ergebnis $u^n$ berechnet und ausgegeben werden. Dazu werden die vorher definierten Methoden benötigt.

- Rechnen Sie mit Bevölkerungsanteilen, d.h. für den Startvektor $u^{[0]}$ gilt $0 \leq x_0, y_0, z_0 \leq 1$ und $x_0 + y_0 + z_0 = 1$. Testen Sie verschiedene Startvektoren und verschiedene Werte für $n$. Welcher stationäre Zustand stellt sich unabhängig vom Startvektor für hinreichend große $n$ ein? Wie hoch ist der Anteil der Bauern?

**Musterlösung:** stationärer Zustand:

$$u_n = \begin{pmatrix} 1/6 \\ 1/3 \\ 1/2 \end{pmatrix}$$

**Fragen 1** *Vektoriteration*

- Nennen Sie zwei iterative Verfahren zum Finden von Nullstellen von Funktionen und erklären Sie diese!

**Aufgabe 2** *Das Newton-Verfahren*

Algorithmen, welche Nullstellen von Funktionen näherungsweise bestimmen, nennt man *Nullstellenverfahren*. Eines der bekanntesten Nullstellenverfahren ist das *Newton-Verfahren*. Mit diesem Verfahren können Näherungswerte für die Nullstelle einer differenzierbaren Funktion $f : \mathbb{R} \to \mathbb{R}$ berechnet werden. Ausgehend von einem Startwert $x_0 \in \mathbb{R}$, wird beim Newton-Verfahren eine Folge von Näherungswerten $x_1, x_2, x_3, \ldots$ für die Nullstelle gemäß

$$x_{k+1} := \Phi_f(x_k) \qquad \text{für } k = 0, 1, 2, \ldots$$

mit

$$\Phi_f(x) := x - \frac{f(x)}{f'(x)}$$

iterativ berechnet. Unter bestimmten Voraussetzungen kann man zeigen, dass diese Folge gegen eine Nullstelle von $f$ konvergiert.
Schreiben Sie ein Java-Programm welches Näherungswerte für die Nullstellen der Funktion $f : \mathbb{R} \to \mathbb{R}$, definiert durch $f(x) := -2 \cdot x^2 + 4$, mit dem Newton-Verfahren berechnet.

- Erstellen Sie dazu eine öffentliche Klasse namens `Newtonverfahren1`. Definieren Sie in dieser Klasse eine öffentliche Klassenmethode namens `funktion`, welche die Funktionswerte der Funktion $f$ berechnet. Die Methode soll einen formalen Parameter vom Gleitkommatyp besitzen, über den eine Stelle $x$ übergeben werden kann. Der zugehörige Funktionswert $f(x)$ soll als Wert ebenfalls vom Gleitkommatyp von der Methode zurück gegeben werden. Definieren Sie in gleicher Weise eine öffentliche Klassenmethode namens `ableitung`, welche die Funktionswerte der Ableitung $f'$ berechnet.

- Definieren Sie nun eine öffentliche Klassenmethode namens `newtonOperator`. Die Methode soll einen Wert vom Gleitkommatyp zurückgeben und einen formalen Parameter vom gleichen Typ besitzen. Realisieren Sie mit dieser Methode die Funktion $\Phi_f$. Verwenden Sie dazu die Methoden `funktion` und `ableitung`.

- Erstellen Sie eine öffentliche Klassenmethode `ausgabe`, ohne Rückgabewert und mit drei formalen Parametern: einen ganzzahligen Parameter und zwei vom Gleitkommatyp. Über diese Parameter sollen der Iterationsindex $k$, die aktuelle Näherung $x_k$ sowie der Funktionswert $f(x_k)$ an die Methode übergeben werden. Die Methode soll alle drei Daten auf der Konsole ausgeben.

- Definieren Sie nun die `main`-Methode des Programms. Lesen Sie in dieser Methode zunächst einen Startwert $x_0$ von der Konsole ein. Implementieren Sie anschließend das Newton-Verfahren. Verwenden Sie hierfür eine **`while`**-Schleife. Die Schleife soll so lange durchlaufen werden, wie

  $$|x_{k+1} - x_k| > \varepsilon \qquad \text{und} \qquad k \leq 100$$

  mit $\varepsilon = 10^{-10}$ gilt. Geben Sie die Ergebnisse $k$, $x_k$ und $f(x_k)$ jedes Iterationsschritts auf der Konsole aus. Verwenden Sie die Methoden `newtonOperator` und `ausgabe`.

  **Hinweis:** Die Betragsfunktion $|\cdot|$ heißt in Java `Math.abs()`.

- Berechnen Sie die mithilfe des Programms die Nullstelle der Funktion $f$. Außerdem soll die Nullstelle der Funktion $g : \mathbb{R} \to \mathbb{R}$, definiert durch $g(x) := e^{-x} - x$, ermittelt werden. Erstellen Sie dazu eine Kopie der Klasse `Newtonverfahren1`. Benennen Sie diese Kopie mit `Newtonverfahren2` und modifizieren Sie sie entsprechend.

**Hinweis:** Die Nullstellen für $f(x)$ sind $\pm\sqrt{2}$ und die für $g(x)$ ist $\approx 0,567143$.

## Aufgabe 3 *Sortieren*

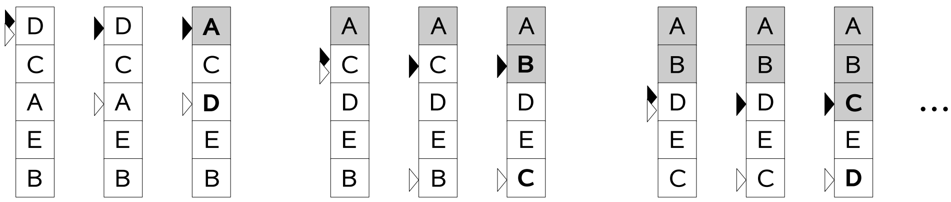
In der elektronischen Datenverarbeitung taucht häufig das Problem auf, eine Folge von Daten bezüglich einer bestimmten Ordnungsrelation zu sortieren. Hierbei kommen sogenannte *Sortieralgorithmen* zum Einsatz. Ein relativ einfacher Sortieralgorithmus ist der *MinSort*–Algorithmus. Er funktioniert folgendermaßen:

(1) Gegeben sei ein Feld der Länge $n$. Der Index der ersten Feldkomponente sei $0$.

(2) Setze $i = 0$.

(3) Solange $i < n - 1$ gilt, bestimme den Index $k$ der kleinsten Feldkomponente mit $i \leq k \leq n - 1$, vertausche die Feldkomponenten mit den Indizes $i$ und $k$ und erhöhe $i$ anschließend um Eins.

In der folgenden Abbildung sind die ersten drei Schritte des *MinSort*–Algorithmus schematisch dargestellt. Das schwarze Dreieck bezeichnet dabei die Feldkomponente mit dem Index $i$, das weiße Dreieck die mit dem Index $k$. Die grau unterlegten Feldkomponenten stellen den *sortierten* Teil des Feldes dar.




Erstellen Sie ein Java–Programm mit dem Namen `MinSort`, welches eine Liste ganzer Zahlen einliest, und diese mit dem *MinSort*–Algorithmus sortiert. Gehen Sie dabei wie folgt vor:

(a) Erstellen Sie eine `main`–Methode, in der Sie eine ganze Zahl $n$ von der Konsole einlesen und diese in einer Variable vom Typ **`int`** speichern. Erzeugen Sie anschließend ein Feld namens `liste` der Länge $n$ vom Typ **`int`**. Lesen Sie mit einer **`for`**–Schleife $n$ ganze Zahlen von der Konsole ein, und speichern Sie diese in den Feldkomponenten des Feldes `liste`. Geben Sie anschließend die Werte der Feldkomponenten der Reihe nach auf der Konsole aus.

(b) Sortieren Sie das Feld `liste` mit dem *MinSort*–Algorithmus. Durchlaufen Sie dazu das Feld mit einer **`for`**–Schleife. Bestimmen Sie für jeden Index $i$ mit einer inneren **`for`**–Schleife den Index $k \geq i$ der kleinsten Feldkomponente im unsortierten Teil des Feldes. Vertauschen Sie anschließend den Wert der $i$-ten Feldkomponente mit dem der $k$-ten.

(c) Geben Sie die Feldkomponenten des sortierten Feldes auf der Konsole aus. Testen Sie Ihr Programm.