Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M.Sc. Stefan Karch
M.Sc. Mariia Sukhova

09.01.2024

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 9

Bearbeitungszeitraum: 15.01.2024 – 26.01.2024

Aufgabe 1 *Finanzplanung*

Im Finanzwesen versteht man unter einer Geldanlage eine Investition von Geldbeträgen, die nach einer gewissen Zeitspanne einen bestimmten Ertrag erzielen soll. Werden mehrere Anlagen getätigt, so spricht man von einem Anlagenportfolio. Ziel dieser Aufgabe ist es, ein objektorientiertes Java-Programm zu erstellen, mit dem die Auswirkung eines Anlagenportfolios über einen Zeitraum von 10 Jahren simuliert werden kann. Dabei wird von stark vereinfachten Annahmen ausgegangen.

(a) Erstellen Sie eine Objektklasse mit dem Namen `Geldanlage`, die eine Geldanlage abstrakt beschreiben soll. Wir gehen davon aus, dass eine Anlage über einen bestimmten Geldbetrag B in einem sogenannten Anfangsjahr t_0 getätigt wird und danach bis zu einem sogenannten Endjahr t_{Ende} läuft. Der Ertrag E der Geldanlage wird durch den sogenannten Zinssatz z bestimmt, d.h. es gilt

$$E = (1+z)^{t_{\text{Ende}}-t_0} B.$$

Definieren Sie in der Klasse `Geldanlage` für das Anfangsjahr t_0 und für das Endjahr t_{Ende} jeweils eine ganzzahlige Instanzvariable. Definieren Sie weiterhin für den Betrag B und für den Zinssatz z jeweils eine Instanzvariable vom Gleitkommatyp. Erstellen Sie einen Konstruktor, dem Werte für t_0, t_{Ende}, B und z übergeben werden können.

(b) Die Auswirkung einer Geldanlage kann durch eine sogenannte Auszahlungsfunktion a beschrieben werden, die für jedes Jahr t die Kosten bzw. Auszahlung $a(t)$ der Geldanlage zurückliefert. Im Anfangsjahr gilt $a(t_0) = -B$, im Endjahr gilt $a(t_{\text{Ende}}) = E$. Für alle übrigen Jahre, d.h. für $t \neq t_0$ und $t \neq t_{\text{Ende}}$, gilt $a(t) = 0$. Erweitern Sie Ihre Objektklasse `Geldanlage` um eine Instanzmethode namens `auszahlung`, die zu einem gegebenen Jahr t (repräsentiert durch einen **`int`**-Wert) den Wert der Auszahlungsfunktion $a(t)$ zurückgibt.

(c) Erstellen Sie ein Java-Programm mit dem Namen `Finanzplanung`. Erstellen Sie in der `main`-Methode ein Feld von `Geldanlage`-Objekten namens `portfolio` mit drei Komponenten. Erzeugen Sie drei Objekte vom Typ `Geldanlage` mit folgenden Parametern:

Anfangsjahr	Endjahr	Betrag	Zinssatz
2	5	100	3.0 %
4	8	200	2.5 %
6	9	150	1.7 %

Speichern Sie diese Objekte in den Feldkomponenten von `portfolio` ab. Berechnen Sie anschließend für die Jahre $1, 2, 3 \ldots, 10$ die jeweilige Summe der Auszahlungsfunktionen der einzelnen Geldanlagen und geben Sie diese auf dem Bildschirm aus. Sie sollten folgendes Ergebnis erhalten

Jahr	Summe der Auszahlungen
1	0.00
2	-100.00
3	0.00
4	-200.00
5	109.27
6	-150.00
7	0.00
8	220.76
9	157.78
10	0.00

Aufgabe 2 *Eigenwertberechnung bei Tridiagonalmatrizen*

Sei $A \in \mathbb{R}^{n \times n}$ eine quadratische Matrix, dann heißt $\lambda \in \mathbb{R}$ Eigenwert von A, wenn ein Vektor $x \in \mathbb{R}^n$, mit $x \neq 0$, existiert, so dass

$$Ax = \lambda x. \tag{1}$$

Eigenwerte spielen in vielen Bereichen, wie z.B. bei der Berechnung von Eigenfrequenzen, eine wichtige Rolle. Den betraglich größten Eigenwert kann man mit der sogenannten Potenzmethode numerisch berechnen. Diese Methode kann als Algorithmus folgendermaßen beschrieben werden:

1. Seien $A \in \mathbb{R}^{n \times n}$ und $x^{(0)} \in \mathbb{R}^n$, $x^{(0)} \neq 0$, gegeben. Setze $k := 0$.
2. Berechne

$$\begin{aligned} q^{(k+1)} &= x^{(k)} / \|x^{(k)}\|_2 \\ x^{(k)} &= A q^{(k+1)} \\ \lambda^{(k+1)} &= \langle q^{(k+1)}, x^{(k+1)} \rangle \\ k &= k + 1 \end{aligned}$$

   solange $\left|\lambda^{(k+1)} - \lambda^{(k)}\right| > \epsilon$.
3. $\lambda^{(k)}$ ist eine Näherung an den betragsmäßig größten Eigenwert.

Dabei ist das Skalarprodukt $\langle\,.\,,\,.\,\rangle$ zwischen zwei Vektoren $x, y \in \mathbb{R}^n$ sowie die Norm $\|\,.\,\|_2$ für einen Vektor $x \in \mathbb{R}^n$ gegeben durch:

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i, \quad \text{bzw.} \quad \|x\|_2 = \sqrt{\langle x, x\rangle}.$$

In der Praxis treten häufig sogenannte Tridiagonalmatrizen auf. Dies sind quadratische Matrizen, die nur Einträge auf der Diagonalen und auf den beiden Nebendiagonalen besitzen. D.h. eine Tridiagonalmatrix $T \in \mathbb{R}^{n\times n}$ besitzt folgende Form:

$$T = \begin{pmatrix} b_0 & c_0 & & & 0 \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & \cdot & \\ & & \cdot & \cdot & c_{n-2} \\ 0 & & & a_{n-1} & b_{n-1} \end{pmatrix}.$$

Um den Speicheraufwand gering zu halten, speichert man nur die Einträge die ungleich Null sind in einer Matrix $C \in \mathbb{R}^{n\times 3}$ ab, die folgende Gestalt besitzt:

$$C = \begin{pmatrix} 0 & b_0 & c_0 \\ a_1 & b_1 & c_1 \\ \cdot & \cdot & \cdot \\ a_{n-2} & b_{n-2} & c_{n-2} \\ a_{n-1} & b_{n-1} & 0 \end{pmatrix}. \tag{2}$$

Schreiben Sie ein Java-Programm, welches den betragsmäßig größten Eigenwert einer Tridiagonalmatrix mithilfe der Potenzmethode berechnet. Gehen Sie dabei folgendermaßen vor:

(a) Erstellen Sie eine Klasse mit dem Namen `TridMatrix`, welche eine Tridiagonalmatrix $T \in \mathbb{R}^{n\times n}$ repräsentiert. Definieren Sie für die Klasse `TridMatrix` die folgenden Elemente: Eine private ganzzahlige Instanzvariable namens `n`, die die Dimension der Tridiagonalmatrix speichert, und eine private Instanzvariable namens `values` vom Typ **`double`**`[][]`, welche die Matrixeinträge entsprechend Gleichung (2) speichert.

(b) Definieren Sie einen öffentlichen Konstruktor mit drei formalen Parameter a, b und c vom Typ **`double`**`[]`, welche die Spalten der Matrix C initialisieren sollen. Weisen Sie im Konstruktor der Instanzvariablen `n` die Länge des übergebenen Feldes b zu. Initialisieren Sie die Instanzvariable `values` mit den Werten in den übergebenen Feldern.

(c) Erstellen Sie eine Klassenmethode namens `skalarprodukt` mit zwei Parameter x und y vom Typ **`double`**`[]` und Rückgabewert vom Gleitkommytyp, die das Skalarprodukt zwischen x und y berechnet und zurückgibt.

(d) Erstellen Sie eine Instanzmethode namens `display` ohne formale Parameter, die die Tridiagonalmatrix formatiert auf der Konsole ausgibt.

(e) Erstellen Sie eine Instanzmethode namens `vtimes` mit einem Parameter b vom Typ **`double`**`[]`, die ein Feld vom Typ **`double`**`[]` zurückgibt. Die Methode soll das Produkt der Tridiagonalmatrix mit dem Vektor b berechnen und zurückgeben.

(f) Erstellen Sie eine Instanzmethode namens `eigenwert` mit einem Parameter $x0$ vom Typ **`double`**`[]`. Diese soll ausgehend vom Startvektor $x0$ den betragsmäßig größten Eigenwert der Tridiagonalmatrix mithilfe der Potenzmethode berechnen. Hinweis: Verwenden Sie hierzu die Klassenmethode `skalarprodukt` sowie die Instanzmethode `vtimes`.

(g) Definieren Sie in einer zweiten öffentlichen Klasse namens `Eigenwertberechnung` die `main`-Methode Ihres Programms. Erstellen Sie drei Felder a, b und c vom Typ **`double`**`[]`, die die Einträge der Tridiagonalmatrix enthalten sollen. Erzeugen Sie ein weiteres Feld $x0$ vom Typ **`double`**`[]` und initialisieren Sie die Einträge mit 1. Erzeugen Sie eine Tridiagonalmatrix mithilfe des Konstruktors und berechnen Sie den betragsmäßig größten Eigenwert mit der Funktion `eigenwert`. Geben Sie auf der Konsole sowohl die Matrix als auch den EIgenwert aus.

Aufgabe 3 (Pflichtaufgabe) *Quaternionen*

Mit $\mathbb{H}$ wird die Menge der *Quaternionen* bezeichnet. Eine Quaternion ist dabei ein Quadrupel

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \in \mathbb{R}^4.$$

Man definiert die *Summe* $p + q$ und das *Produkt* $p \cdot q$ zweier Quaternionen $p, q \in \mathbb{H}$ durch

$$p + q := \begin{bmatrix} p_0 + q_0 \\ p_1 + q_1 \\ p_2 + q_2 \\ p_3 + q_3 \end{bmatrix} \quad \text{und} \quad p \cdot q := \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 + p_2q_0 + p_3q_1 - p_1q_3 \\ p_0q_3 + p_3q_0 + p_1q_2 - p_2q_1 \end{bmatrix}.$$

Ferner definiert man den *Betrag* $|p|$ einer Quaternion $p \in \mathbb{H}$ durch

$$|q| := \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}.$$

Quaternionen werden zum Beispiel in der Robotik zur Beschreibung und effizienten Darstellung von Drehungen von Roboterarmen eingesetzt. Aber auch in vielen Bereichen der Simulation anderer bewegter Körper finden sie Anwendung. Erstellen Sie nun ein Java-Programm, welches das Rechnen mit Quaternionen realisiert. Gehen Sie dabei folgendermaßen vor:

(a) Erstellen Sie eine öffentliche Klasse namens `Quaternion` mit vier privaten Instanzvariablen vom Gleitkommatyp für die Komponenten q_0, q_1, q_2, q_3 einer Quaternion q. Definieren Sie für die Klasse außerdem einen öffentlichen Konstruktur mit vier Parametern vom entsprechenden Typ, der den Instanzvariablen die entsprechenden Parameterwerte zuweist.

(b) Definieren Sie für die Klasse *Quaternion* eine öffentliche Klassenmethode namens `einlesen` ohne Parameter. Die Methode soll vier Gleitkomma-Komponenten einer Quaternion von der Konsole einlesen und anschließend eine entsprechende Instanz der Klasse `Quaternion` zurück geben.

(c) Definieren Sie für die Klasse *Quaternion* eine öffentliche Instanzmethode namens `betrag` ohne Parameter, die den Betrag der Instanz berechnet, und diesen als Wert vom Gleitkommatyp zurück gibt.

(d) Definieren Sie für die Klasse *Quaternion* zwei öffentliche Klassenmethoden namens `summe` und `prod`, mit jeweils zwei Parametern vom Typ `Quaternion`. Die Methoden sollen die Summe bzw. das Produkt der übergebenen Instanzen berechnen und als Instanz der Klasse `Quaternion` zurück geben.

(e) Definieren Sie für die Klasse *Quaternion* eine öffentliche Instanzmethode namens `toString` ohne Parameter, die eine Darstellung der Quaternion von der Form „$[q_0; q_1; q_2; q_3]$“als Zeichenkette zurück gibt.

(f) Erstellen Sie eine zweite öffentliche Klasse namens `Quaternionen` mit einer `main`-Methode. Lesen Sie in der `main`-Methode zwei Quaternionen von der Konsole ein. Berechnen Sie die Summe und das Produkt der eingelesenen Quaternionen, und geben Sie diese zusammen mit den Beträgen der eingelesenen Quaternionen auf der Konsole aus. Verwenden Sie, soweit möglich, die Instanz- und Klassenmethoden der Klasse `Quaternion`.

Fragen 3 *Quaternionen*

- Was ist der Unterschied zwischen einer Klasse und einer Instanz?
- Was ist der Vorteil von objektorientierter Programmierung?