

## Übungen zur Vorlesung Computermathematik

### Serie 3

**Aufgabe 3.1.** Das  $\Delta^2$ -Verfahren von Aitken ist ein Verfahren zur Konvergenzbeschleunigung von Folgen. Für eine injektive Folge  $(x_n)$  mit  $x = \lim_{n \rightarrow \infty} x_n$  definiert man

$$y_n := x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$$

Unter gewissen Voraussetzungen an die Folge  $(x_n)$  gilt dann

$$\lim_{n \rightarrow \infty} \frac{x - y_n}{x - x_n} = 0,$$

d.h. die Folge  $(y_n)$  konvergiert schneller gegen  $x$  als  $(x_n)$ . Schreiben Sie eine MATLAB-Funktion `aitken`, die einen Vektor  $x \in \mathbb{R}^N$  übernimmt und den Vektor  $y \in \mathbb{R}^{N-2}$  zurückgibt. Verwenden Sie dazu geeignete Schleifen.

**Aufgabe 3.2.** Schreiben Sie eine alternative MATLAB-Funktion `aitken_vec`, die den Vektor  $y \in \mathbb{R}^{N-2}$  aus Aufgabe 3.1 mittels geeigneter Vektor-Arithmetik statt Schleifen berechnet.

**Aufgabe 3.3.** Erweitern Sie den Code von Folie 106, indem Sie das Konvergenzverhalten des einseitigen und zentralen Differenzenquotientens mit dem Aitkenschen  $\Delta^2$ -Verfahren verbessern. Welche Konvergenzraten beobachten Sie? Visualisieren Sie diese geeignet!

**Aufgabe 3.4.** Schreiben Sie eine Funktion `diffaitken`, die die Ableitung einer Funktion in einem Punkt  $x$  approximiert. Dazu verwende man den zentralen Differenzenquotienten

$$\Phi(h) = \frac{f(x+h) - f(x-h)}{2h}.$$

Für gegebene Startschrittweite  $h_0 > 0$  und Toleranz  $\tau$  berechne man die Folgen  $h_n := 2^{-(n-1)}h_0$ ,  $x_n := \Phi(h_n)$ , und  $\phi_n := x_n$  für  $n = 1, 2$  bzw. sei  $\phi_n := y_{n-2}$  für  $n \geq 3$  der Wert der Aitken-Extrapolierten. Die Iteration werde beendet, falls  $n \geq 2$  und

$$|\phi_n - \phi_{n-1}| \leq \begin{cases} \tau & \text{falls } |\phi_n| \leq \tau, \\ \tau|\phi_n| & \text{sonst.} \end{cases}$$

In diesem Fall werde  $\phi_n$  als Approximation der Ableitung zurückgegeben.

**Aufgabe 3.5.** Sei  $f : [a, b] \rightarrow \mathbb{R}$  eine stetige Funktion. Für  $N \in \mathbb{N}$  und  $x_j := a + j(b - a)/N$  mit  $j = 0, \dots, N$ , definieren wir die *zusammengesetzte Mittelpunktsregel*

$$I_N := \frac{b-a}{N} \sum_{j=1}^N f((x_{j-1} + x_j)/2). \quad (1)$$

Da  $I_N$  eine Riemannsumme ist, wissen wir, dass

$$\lim_{N \rightarrow \infty} I_N = \int_a^b f dx.$$

Für  $f \in C^2[a, b]$ , kann man sogar zeigen, dass

$$\left| \int_a^b f dx - I_N \right| = \mathcal{O}(N^{-2}).$$

Schreiben Sie eine MATLAB Funktion

```
int = midpointrule(a,b,f,n),
```

welche für die Folge  $N = 2^k$  und  $k = 0, \dots, n$ , den Vektor `int` der entsprechenden Werte  $I_N$  berechnet und zurückgibt.

**Aufgabe 3.6.** Betrachten Sie das Integral  $I := \int_0^5 \exp(x) dx$ . Verwenden Sie die zusammengesetzte Mittelpunktsregel aus Aufgabe 3.5, um die Folge der approximativen Integrale  $I_N$  zu berechnen. Verwenden Sie einen doppelt logarithmischen Plot, um den Fehler  $E_N = |I - I_N|$  und den Fehlerschätzer  $\delta_N = |I_{2N} - I_N|$  zu visualisieren. Verifizieren Sie das Konvergenzverhalten  $\mathcal{O}(N^{-2})$ . Welches Konvergenzverhalten beobachten Sie, wenn Sie das  $\Delta^2$ -Verfahren von Aitken verwenden? Welches Konvergenzverhalten beobachten Sie, wenn Sie die Auswertung  $f((x_{j-1} + x_j)/2)$  beim Mittelpunkt durch die Auswertung  $f(x_{j-1})$  ersetzen?

**Aufgabe 3.7.** Alternativ zum Bisektionsverfahren aus der Vorlesung kann man zur Berechnung einer Nullstelle einer Funktion  $f : [a, b] \rightarrow \mathbb{R}$  das *Newton-Verfahren* verwenden. Ausgehend von einem Startwert  $x_0$  definiert man induktiv eine Folge  $(x_n)$  wie folgt: Zu gegebenem  $x_k$  sei  $x_{k+1}$  die Nullstelle der Tangente an den Graphen von  $f$  im Punkt  $(x_k, f(x_k))$ , d.h.  $x = x_{k+1}$  erfüllt  $0 = f(x_k) + f'(x_k)(x - x_k)$ . Auflösen nach  $x$  zeigt

$$x_{k+1} = x_k - f(x_k)/f'(x_k).$$

Realisieren Sie das Newton-Verfahren in einer Funktion `newton(f, fprime, x0, tau)`, wobei die Iteration abgebrochen wird, falls entweder

$$|f'(x_n)| \leq \tau$$

oder

$$|f(x_n)| \leq \tau \quad \text{und} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{für } |x_n| \leq \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. In jedem Fall werde  $x_n$  als Approximation der gesuchten Nullstelle zurückgeben, wobei im ersten Fall zusätzlich eine Warnung ausgegeben werden soll.– Neben  $x_n$  sollen die Folgen  $(x_0, \dots, x_n)$  der approximativen Nullstellen und der dazugehörigen Funktionswerte zurückgegeben werden. Testen Sie Ihre Implementierung mit der Funktion  $f(x) = x^2 + e^x - 2$ .

**Aufgabe 3.8.** Die sogenannte *Power-Iteration* approximiert (unter gewissen Voraussetzungen) den betragsgrößten Eigenwert  $\lambda \in \mathbb{R}$  einer symmetrischen Matrix  $A \in \mathbb{R}^{n \times n}$  sowie einen dazugehörigen Eigenvektor  $x \in \mathbb{R}^n$ . Dazu wählt man einen Startvektor  $x^{(0)} \in \mathbb{R}^n \setminus \{0\}$ , z.B.  $x^{(0)} = (1, \dots, 1) \in \mathbb{R}^n$ . Man definiert induktiv für  $k \in \mathbb{N}$  die Folgen

$$x^{(k)} := \frac{Ax^{(k-1)}}{\|Ax^{(k-1)}\|_2} \quad \text{und} \quad \lambda_k := x^{(k)} \cdot Ax^{(k)} := \sum_{j=1}^n x_j^{(k)} (Ax^{(k)})_j,$$

wobei  $\|y\|_2 := (\sum_{j=1}^n y_j^2)^{1/2}$  die euklidische Norm bezeichne. Dann konvergiert die Folge  $(\lambda_k)$  gegen  $\lambda$  und  $(x^{(k)})$  konvergiert (in einem geeigneten Sinn) gegen einen Eigenvektor zu  $\lambda$ . Schreiben Sie eine Funktion `poweriteration`, die eine Matrix  $A$ , eine Toleranz  $\tau$  und einen Startvektor  $x^{(0)}$  übernimmt, dann  $A$  auf Symmetrie überprüft und ggf. mit Fehlermeldung abbricht und schließlich die Folgen  $(\lambda_k)$  und  $(x^{(k)})$  berechnet, bis gilt

$$\|Ax^{(k)} - \lambda_k x^{(k)}\|_2 \leq \tau \quad \text{und} \quad |\lambda_{k-1} - \lambda_k| \leq \begin{cases} \tau & \text{für } |\lambda_k| \leq \tau, \\ \tau |\lambda_k| & \text{sonst.} \end{cases}$$

Die Funktion liefere in diesem Fall  $\lambda_k$  und  $x^{(k)}$  zurück. Realisieren Sie die Funktion möglichst rechenökonomisch, d.h. vermeiden Sie unnötige Berechnungen (insb. von Matrix-Vektor-Produkten), indem Sie Ergebnisse ggf. zwischenspeichern. Sie können Ihre Funktion mit Hilfe der MATLAB-Funktion `eig` verifizieren. Verwenden Sie die Funktion `norm` sowie MATLAB-Arithmetik, soweit wie möglich.