Dirk Praetorius
Marcus Page

Sommersemester 2011
08. Juni 2011

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1*. Für eine differenzierbare Funktion $f : [a, b] \to \mathbb{R}$ kann man die Ableitung $f'(x)$ in einem festen Punkt $x \in \mathbb{R}$ durch den einseitigen Differenzenquotienten

$$\Phi(h) := \frac{f(x+h) - f(x)}{h} \quad \text{für } h > 0$$

approximieren. Nach Taylor-Formel gilt für $f \in C^2(\mathbb{R})$ die Fehlerabschätzung $|f'(x) - \Phi(h)| = \mathcal{O}(h)$. Beweisen Sie diese Aussage mathematisch! Schreiben Sie eine Funktion `diff(f,x,h0,tau)`, die für $h_n := 2^{-n} h_0$ die Folge der $\Phi(h_n)$ berechnet, bis gilt

$$|\Phi(h_n) - \Phi(h_{n+1})| \leq \begin{cases} \tau & \text{für } |\Phi(h_n)| \leq \tau, \\ \tau\,|\Phi(h_n)| & \text{anderenfalls.} \end{cases}$$

Die Funktion liefere in diesem Fall die vollständige Folge $(\Phi(h_0), \dots, \Phi(h_n))$ der Iterierten zurück. Speichern Sie den Source-Code unter `diff.m` in das Verzeichnis `serie11`.

Aufgabe 11.2*. Man kombiniere das Aitken-Verfahren aus Aufgabe 8.3 mit dem einseitigen Differenzenquotienten $\Phi(h)$ aus Aufgabe 10.1: Mit $h_n := 2^{-n} h_0$ betrachten wir die Folge der $x_n := \Phi(h_n)$ und erhalten daraus die Folge (y_n). Man schreibe eine Funktion `diffaitken`, die neben dem Funktionspointer einer Funktion f den Auswertungspunkt x, die Schrittweite $h_0 > 0$ sowie die Toleranz $\tau > 0$ übernimmt und $y_{n+1} \approx f'(x)$ zurückliefert, sobald gilt

$$|y_n - y_{n+1}| \leq \begin{cases} \tau, & \text{falls } |y_{n+1}| \leq \tau, \\ \tau\,|y_{n+1}|, & \text{anderenfalls.} \end{cases}$$

In jedem Schritt gebe man h, $|y_{n+1} - y_n|$ sowie y_{n+1} aus. Als Beispiel betrachte man die Berechnung von $e = \exp(1) = \exp'(1)$ und $\varepsilon = 10^{-12}$. Vergleichen Sie die Anzahl der Iterationen mit und ohne (d.h. $y_n = x_n$) Aitken-Verfahren. Speichern Sie den Source-Code unter `diffaitken.c` in das Verzeichnis `serie11`.

Aufgabe 11.3*. Eine effiziente Implementierung des einseitigen Differenzenquotienten $\Phi(h)$ aus Aufgabe 11.1 verwendet die vorherigen Werte $\Phi(h_0), \dots, \Phi(h_n)$, indem man (theoretisch!) das Interpolationspolynom p_n vom Grad $n-1$ zu den Punkten $(h_j, \Phi(h_j))$ für $j = 1, \dots, n$ betrachtet, d.h. $p_n(h) \approx \Phi(h)$, und dieses mit dem Neville-Verfahren bei $h = 0$ auswertet. Man bezeichnet dieses Vorgehen als *Richardson-Extrapolation des einseitigen Differenzenquotienten.* (Einen Konvergenzbeweis für dieses Verfahren sehen Sie in der Vorlesung zur Numerischen Mathematik.) Mit $h_n := 2^{-n} h_0$ betrachten wir die Folge der $y_n := p_n(0)$. Schreiben Sie eine Funktion `richardson`, die neben dem Funktionshandle einer Funktion f, den Auswertungspunkt x, die erste Schrittweite $h_0 > 0$ sowie die Toleranz $\tau > 0$ übernimmt und $y_{n+1} \approx f'(x)$ zurückliefert, sobald gilt

$$|y_n - y_{n+1}| \leq \begin{cases} \tau, & \text{falls } |y_{n+1}| \leq \tau, \\ \tau\,|y_{n+1}| & \text{anderenfalls.} \end{cases}$$

Verwenden Sie bei der Realisierung die Funktion `neville` aus Aufgabe 9.1 bzw. Aufgabe 9.2. Speichern Sie den Source-Code unter `richardson.m` in das Verzeichnis `serie11`.

Aufgabe 11.4*. Bei der Richardson-Extrapolation des einseitigen Differenzenquotienten aus Aufgabe 11.3 kann man sich folgende Beobachtung zum Neville-Verfahren zu Nutze machen: Bei einer effizienten Implementierung des Neville-Verfahrens gemäss Aufgabe 9.2 überschreibt man den Vektor y durch die Diagonale $(p_{1,n}, p_{2,n-1}, \ldots, p_{n,1})$, und $p_{1,n}$ ist der gesuchte Wert. Fügt man nun einen weiteren Interpolationsknoten (x_{n+1}, y_{n+1}) hinzu, so muss man nicht noch einmal das vollständige Schema rechnen, sondern lediglich die „neue Diagonale“ $(p_{1,n+1}, p_{2,n}, \ldots, p_{n+1,1})$. Mit dieser Beobachtung muss man in jedem Schritt der Richardson-Extrapolation nur noch eine Schleife durchlaufen, nicht mehr zwei! Realisieren Sie dieses Vorgehen in einer Funktion `richardson2`, d.h. binden Sie den Code aus Aufgabe 9.2 geeignet ein. Speichern Sie den Source-Code unter `richardson2.m` in das Verzeichnis `serie11`.

Aufgabe 11.5. Das Integral $\int_a^b f\,dx$ einer stetigen Funktion f wird üblicherweise durch sogenannte Quadraturformeln berechnet. Die *Trapezregel* $I_1(f,a,b)$ ist beispielsweise durch

$$I_1(f,a,b) := \frac{b-a}{2}\big(f(a)+f(b)\big)$$

gegeben. Schreiben Sie eine rekursive Funktion `adquad`, die das Integral $\int_a^b f\,dx$ mittels Trapezregel $I_1(f,a,b)$ und iterierter Intervallhalbierung

$$\int_a^b f\,dx = \int_a^{(a+b)/2} f\,dx + \int_{(a+b)/2}^b f\,dx \approx I_1(f,a,(a+b)/2) + I_1(f,(a+b)/2,b)) =: I_1^\star(f,a,b)$$

berechnet. Dabei soll der Integrationsbereich jeweils halbiert werden, falls

$$|I_1(f,a,b) - I_1^\star(f,a,b)| > \varepsilon \max\{|I_1(f,a,b)|, |I_1^\star(f,a,b)|\} \quad \text{und} \quad |b-a| > h_{\min}$$

ist. Der Funktion sollen neben f, a und b auch die Parameter ε und $h_{\min}$ übergeben werden. Vermeiden Sie weitestgehend unnötige f-Auswertungen. Sie können Ihren Code mit Hilfe der Matlab-Funktion `quad` verifizieren.

Aufgabe 11.6. Erweitern Sie den Code aus Aufgabe 11.5 so, dass die Anzahl der Funktionsauswertungen gezählt wird. Schreiben Sie einen weiteren Code, bei dem Sie $I_1^\star(f,a,b)$ durch die *Simpson-Regel*

$$I_2(f,a,b) := \frac{b-a}{6}\Big(f(a) + 4\,f\big(\frac{a+b}{2}\big) + f(b)\Big)$$

ersetzen. Vergleichen Sie beide Implementierungen anhand der Beispiele $\int_0^x \exp(t)\,dt = \exp(x) - 1$ und $\int_0^x \log(t)\,dt = x(\log(x) - 1)$. Plotten Sie mittels `semilogy` jeweils für gewisse Toleranzen $\varepsilon = 10^{-2}, \ldots, 10^{-12}$ den tatsächlichen Fehler $|\text{exakt} - \text{approx}|$ auf der y-Achse über die Anzahl der benötigten Funktionsauswertungen auf der x-Achse.

Aufgabe 11.7. Gegeben seien eine Matrix $A \in \mathbb{R}^{n\times n}$ und eine rechte Seite $b \in \mathbb{R}^n$. Lösen Sie das Gleichungssystem $Ax = b$ mit dem *Gauss'schen Eliminationsverfahren*. Dies ist gerade das Vorgehen, wenn man ein lineares Gleichungssystem händisch löst:

- Zunächst bringt man die Matrix A auf obere Dreiecksform, indem man die Unbekannten eliminiert. Gleichzeitig modifiziert man die rechte Seite b.
- Das entstandene Gleichungssystem mit oberer Dreiecksmatrix A löst man mit Aufgabe 8.7

Im ersten Eliminationsschritt zieht man geeignete Vielfache der ersten Zeile von den übrigen Zeilen ab und erhält dadurch eine Matrix der Form

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

Im zweiten Eliminationsschritt zieht man nun geeignete Vielfache der zweiten Zeile von den übrigen Zeilen ab und erhält eine Matrix der Form

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3} & \dots & a_{nn} \end{pmatrix},$$

Nach $n-1$ Eliminationsschritten erhält man also eine obere Dreiecksmatrix A. Berücksichtigen Sie, dass auch die rechte Seite $b \in \mathbb{R}^n$ geeignet modifiziert werden muss und machen Sie sich das Vorgehen zunächst an einem Beispiel mit $A \in \mathbb{R}^{2\times 2}$ sowie $A \in \mathbb{R}^{3\times 3}$ klar. Schreiben Sie eine Funktion `gauss`, die die Lösung von $Ax = b$ berechnet.

Aufgabe 11.8. Das Gauss'sche Eliminationsverfahren scheitert, falls im k-ten Schritt $a_{kk} = 0$ gilt, auch wenn das Gleichungssystem $Ax = b$ eine eindeutige Lösung x besitzt. Deshalb kann man das Verfahren um eine sogenannte *Pivot-Suche* erweitern:

- Im k-ten Schritt wählt man aus $a_{kk}, \dots, a_{nk}$ das betragsgrösste Element a_{pk}.
- Dann vertauscht man die k-te und die p-te Zeile von A (und b).
- Schliesslich führt man den Eliminationsschritt aus wie zuvor.

Schreiben Sie eine Funktion `gausspivot`, die die Lösung von $Ax = b$ wie angegeben berechnet. (Man kann übrigens mathematisch beweisen, dass das Gauss-Verfahren mit Pivot-Suche genau dann durchführbar ist, wenn das Gleichungssystem $Ax = b$ eine eindeutige Lösung besitzt. Einen Beweis dazu sehen Sie in der Vorlesung zur Numerischen Mathematik.)

Aufgabe 11.9. Wenn man im Gauss-Verfahren mit Pivot-Suche die Zeilen im Eliminationsschritt *wirklich* vertauscht (d.h. Speicher kopiert), führt dies auf unnötig viele Operationen und entsprechend lange Laufzeit des Programms. Es empfiehlt sich daher, die Vertauschung nur *virtuell* durchzuführen: Man startet mit einem Buchhaltervektor $\pi = (1, \dots, n)$. Im Vertauschungsschritt vertauscht man lediglich $\pi(p)$ mit $\pi(k)$. Im Source-Code (sowohl zu Aufgabe 8.7 als auch zu Aufgabe 11.7) sind jetzt die Zeilenindizes, d.h. der erste Index von a_{jk} sowie der Index von b_j geeignet zu modifizieren.

Aufgabe 11.10. Schreiben Sie eine `mex`-Funktion, der Sie eine komplexwertige Matrix aus Matlab übergeben. Die Funktion soll dann den Realteil der Matrix auslesen und als reellwertige Matrix zurückgeben. Ausserdem soll der Imaginärteil der Matrix noch verdreifacht werden. Auch diese Änderung geben Sie als (komplexwertige) Matrix zurück. Überprüfen Sie beide Matrizen in Matlab.