Samuel Ferraz-Leite, Dirk Praetorius
Christoph Überhuber

Sommersemester 2008
28. Mai 2008

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Die Aufgaben mit Stern (*) sind bis zur nächsten Übung vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und mir als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code in ein Unterverzeichnis `serie09` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit* `matlab` *auf der* `cad.zserv.tuwien.ac.at` *interpretiert werden können.*

Aufgabe 81*. Zu gegebenen reellen Stützstellen $x_1 < \cdots < x_n$ und Funktionswerten $y_j \in \mathbb{R}$ garantiert die Lineare Algebra ein eindeutiges Polynom $p(t) = \sum_{j=1}^n a_j t^{j-1}$ vom Grad $n-1$ mit $p(x_j) = y_j$ für alle $j = 1, \dots, n$. Nun sei $t \in \mathbb{R}$ fixiert und $p(t)$ gesucht. Man kann $p(t)$ mit dem *Neville-Verfahren* berechnen, ohne zunächst den Koeffizientenvektor $a \in \mathbb{R}^n$ berechnen zu müssen: Dazu definiere man für $j, m \in \mathbb{N}$ mit $m \geq 2$ und $j + m \leq n+1$ die Werte

$$
\begin{aligned}
p_{j,1} &:= y_j, \\
p_{j,m} &:= \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}.
\end{aligned}
$$

Es gilt dann $p(t) = p_{1,n}$. Man schreibe eine Funktion `neville`, die den Auswertungspunkt $t \in \mathbb{R}$ sowie die Vektoren $x, y \in \mathbb{R}^n$ übernimmt und $p(t)$ mittels Neville-Verfahren berechnet. Dazu berücksichtige man das folgende schematische Vorgehen

$$
\begin{array}{ccccccccccccc}
y_1 & = & p_{1,1} & \longrightarrow & p_{1,2} & \longrightarrow & p_{1,3} & \longrightarrow & \cdots & \longrightarrow & p_{1,n} & = & p(t) \\
 & & & \nearrow & & \nearrow & & & & \nearrow & & & \\
y_2 & = & p_{2,1} & \longrightarrow & p_{2,2} & & & & & & & & \\
 & & & \nearrow & & & & \nearrow & & & & & \\
y_3 & = & p_{3,1} & \longrightarrow & \vdots & & & & & & & & \\
\vdots & & \vdots & & \vdots & \nearrow & & & & & & & \\
y_{n-1} & = & p_{n-1,1} & \longrightarrow & p_{n-1,2} & & & & & & & & \\
 & & & \nearrow & & & & & & & & & \\
y_n & = & p_{n,1} & & & & & & & & & &
\end{array} \tag{1}
$$

Der mathematische Beweis für diesen Algorithmus folgt in der Vorlesung zur Numerischen Mathematik. Zunächst schreibe man die Funktion so, dass die Matrix $(p_{j,m})_{j,m=1}^n$ vollständig aufgebaut wird. Sie können den Code testen, indem Sie für ein bekanntes Polynom p als Funktionswerte $y_j = p(x_j)$ wählen.

Aufgabe 82*. Man kann das Neville-Verfahren aus Aufgabe 81 so programmieren, dass zur Speicherung der Werte *keine* Matrix $(p_{j,m})_{j,m=1}^n$ aufgebaut wird, sondern die gegebenen y_j-Werte geeignet überschrieben werden. Dadurch wird kein weiterer Speicher benötigt. Man realisiere dieses Vorgehen in einer Funktion `neville2`.

Aufgabe 83*. Die Sinus-Funktion hat die Reihendarstellung

$$\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}.$$

Wir betrachten die Partialsummen

$$S_n(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!}.$$

Man schreibe eine Funktion `sin_`, die für gegebene $x \in \mathbb{R}$ und $\tau > 0$ den Wert $S_n(x)$ zurückliefert, sobald

$$|S_n(x) - S_{n-1}(x)|/|S_n(x)| \leq \tau \quad \text{oder} \quad |S_n(x)| \leq \tau$$

gilt. Schreiben Sie ein Matlab-Skript, in dem neben dem berechneten Wert $S_n(x)$ auch der korrekte Wert $\sin(x)$ und der absolute Fehler $|S_n(x) - \sin(x)|$ sowie der relative Fehler $|S_n(x) - \sin(x)|/|\sin(x)|$ im Fall $\sin(x) \neq 0$ ausgegeben werden.

Aufgabe 84*. Realisieren Sie das Newton-Verfahren als MATLAB-Funktion `newton(f,fprime,x0,tau)`: Ausgehend von einem Startwert x_0 definiert man induktiv

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

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. Im ersten Fall werde mit Fehlermeldung abgebrochen (siehe `help error`). Die Parameter `f` und `fprime` seien Funktionshandle der Funktionen f und f'. Testen Sie Ihre Implementierung am Beispiel von $f(t) = 2 - t^2$.

Aufgabe 85. Man schreibe die Funktion in Aufgabe 83 so, dass diese mit einer Schleife auskommt und dass x^{2k+1} und $(2k+1)!$ möglichst kostensparend realisiert werden. Man verwende also insbesondere keine (vor- oder selbst implementierten) Funktionen zur Berechnung der Potenz oder der Faktoriellen.

Aufgabe 86. 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 $|f'(x) - \Phi(h)| = \mathcal{O}(h)$. 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 $(\Phi(h_0), \ldots, \Phi(h_n))$ der Iterierten zurück.

Aufgabe 87. Eine effiziente Implementierung des einseitigen Differenzenquotienten $\Phi(h)$ aus Aufgabe 86 verwendet die vorherigen Werte $\Phi(h_0), \ldots, \Phi(h_n)$, indem man (theoretisch!) das Interpolationspolynom p_n vom Grad $n-1$ zu den Punkten $(h_j, \Phi(h_j))$ für $j = 1, \ldots, n$ 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 $y_n := p_n(0)$. Man schreibe 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 81.

Aufgabe 88. Bei der Richardson-Extrapolation des einseitigen Differenzenquotienten aus Aufgabe 87 kann man sich folgende Beobachtung zum Neville-Verfahren zu Nutze machen: Bei einer effizienten Implementierung des Neville-Verfahrens gemäß Aufgabe 82 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!

Aufgabe 89. Für einen stetigen Integranden $f : [a, b] \to \mathbb{R}$ berechnet man das Integral $I := \int_a^b f\,dx$ numerisch über geeignete Summen. Bei der *summierten Trapezregel* berechnet man für gegebenes $n \in \mathbb{N}$ und $h := (b-a)/n$ zum Beispiel

$$I_n := \frac{h}{2}\Big(f(a) + 2\sum_{j=1}^{n-1} f(a+jh) + f(b)\Big). \tag{2}$$

Dies ist gerade das Integral über die stetige und stückweise affine Funktion p mit $p(a + jh) = f(a + jh)$. Man schreibe eine Funktion `trapezregel(f,a,b,tau)`, die die Folge der Approximationen I_n berechnet, bis gilt

$$|I_n - I_{n-1}| \leq \begin{cases} \tau & \text{für } |I_n| \leq \tau, \\ \tau\,|I_n| & \text{anderenfalls.} \end{cases}$$

In diesem Fall gebe man die vollständige Folge $(I_1, \ldots, I_n)$ der Approximationen zurück. Man teste die numerische Integration am Beispiel $f(x) = \exp(x)$ auf dem Intervall $[0, 10]$ und gebe abhängig von n neben den Fehler $|I - I_n|$ tabellarisch aus.

Aufgabe 90. Man schreibe eine Funktion `kalender(jahr1)`, die das nächste Jahr `jahr2` > `jahr1` berechnet, an dem man einen Kalender von Jahr `jahr1` vollständig wiederverwenden kann, d.h. die Wochentage der Jahre `jahr1` und `jahr2` stimmen vollständig überein.