Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Sommersemester 2010
01. Juni 2010

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten. Kopieren Sie die Source-Codes vor der Übung auf Ihren Account auf der `lva.student.tuwien.ac.at` *und überprüfen Sie, ob diese von* `matlab` *auf der* `lva.student.tuwien.ac.at` *interpretiert werden können.*

Aufgabe 10.1*. 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 definiert 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 $p(t) = p_{1,n}$. Schreiben Sie 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. Berücksichtigen Sie 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. Schreiben Sie 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. Speichern Sie den Source-Code im Verzeichnis `serie10` unter dem Namen `neville.m`.

Aufgabe 10.2*. Es sei $L \in \mathbb{R}^{n \times n}$ eine untere Dreiecksmatrix mit $\ell_{jj} \neq 0$ für alle $j = 1, \dots, n$. Dann ist L invertierbar, und die Inverse lässt sich wie folgt rekursiv berechnen: Wir schreiben L in Block-Form

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}$$

mit $L_{11} \in \mathbb{R}^{p \times p}$, $L_{21} \in \mathbb{R}^{q \times p}$ und $L_{22} \in \mathbb{R}^{q \times q}$, wobei $n = p + q$ gilt. Man beachte, dass L_{11} und L_{22} wieder untere Dreiecksmatrizen sind mit $\ell_{jj} \neq 0$. Üblicherweise wählt man $p = n/2$, falls n gerade ist, bzw. $p = (n-1)/2$, falls n ungerade ist. Offensichtlich wird die Inverse L^{-1} dann durch

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1} L_{21} L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}$$

gegeben. Schreiben Sie eine Funktion `invertL`, die die Inverse L^{-1} rekursiv berechnet. Sie können die Korrektheit Ihrer Funktion mit Hilfe der Matlab-Funktion `inv` überprüfen. Speichern Sie den Source-Code im Verzeichnis `serie10` unter dem Namen `invertL.m`.

Aufgabe 10.3*. Man schreibe eine Funktion `pascal(k)`, die die ersten k-Stufen des Pascal'schen Dreiecks ausgibt: Jede Zeile dieses Schemas beginnt und endet mit 1. Die restlichen Zahlen werden als Summe nebeneinanderstehender Zahlen der vorhergegangenen Zeile gebildet. Für $k = 5$ gilt beispielsweise

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Realisieren Sie das Pascalsche Dreieck möglichst rechenökonomisch, insbesondere ohne die Darstellung der Einträge über den Binomialkoeffizienten.

Aufgabe 10.4*. Die Quotientenfolge $(a_{n+1}/a_n)_{n\in\mathbb{N}}$ zur Fibonacci-Folge $(a_n)_{n\in\mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \geq 2,$$

konvergiert gegen den goldenen Schnitt $(1 + \sqrt{5})/2$. Insbesondere konvergiert die Differenz

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

gegen Null. Schreiben Sie eine Funktion `cauchy`, die zu gegebenem $k \in \mathbb{N}$ die kleinste Zahl $n \in \mathbb{N}$ mit $|b_n| \leq 1/k$ zurückgibt. Speichern Sie den Source-Code im Verzeichnis `serie10` unter dem Namen `cauchy.m`.

Aufgabe 10.5. Schreiben Sie eine rekursive Funktion `detlaplace`, die die Determinante $\det(A)$ einer Matrix $A \in \mathbb{R}^{n\times n}$ mit Hilfe des Laplaceschen Entwicklungssatzes berechnet. Sie können Ihre Funktion mit der MATLAB-Funktion `det` verifizieren.

Aufgabe 10.6. Alternativ kann man die Determinante einer Matrix $A \in \mathbb{R}^{n\times n}$ über die normalisierte LU-Zerlegung aus Aufgabe 9.8 berechnen. Es gilt nämlich $\det(A) = \det(L)\det(U) = \det(U) = \prod_{j=1}^{n} u_{jj}$. Schreiben Sie eine Funktion `detLU`, die die Determinante einer Matrix A mittels der normalisierten LU-Zerlegung berechnet. Sie können Ihre Funktion mit der Matlab-Funktion `det` verifizieren. Die Berechnung der LU-Zerlegung können sie in Matlab mittels `lu` überprüfen.

Aufgabe 10.7. Man kann das Neville-Verfahren aus Aufgabe 10.1 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. Realisieren Sie dieses Vorgehen in einer Funktion `neville2`.

Aufgabe 10.8. Schreiben Sie 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.

Aufgabe 10.9. Es sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische und positv definite Matrix, d.h. $Ax\cdot x > 0$ für $x \neq 0$. Dann existiert eine eindeutige untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit Diagonalelementen $\ell_{jj} > 0$ für alle $j = 1\ldots, n$ und $A = LL^T$, d.h. A besitzt eine spezielle LU-Zerlegung mit $U = L^T$. Man bezeichnet diese Faktorisierung als *Cholesky-Faktorisierung*. Leiten Sie aus der Formel für das Matrix-Matrix-Produkt $A = LL^T$ eine Formel für die Einträge von L her, und schreiben Sie eine Funktion `cholesky`, die L zurückliefert. Sie können Ihre Funktion in Matlab mittels `chol` verifizieren.

Aufgabe 10.10. Das Neville-Verfahren aus Aufgabe 10.1 läßt sich rekursiv implementieren. Schreiben Sie eine rekursive Funktion `rekneville`, die die Auswertung $p_{j,m}(t)$ des Neville-Polynoms $p_{j,m}$ rekursiv berechnet. Ist Rekursion in diesem Beispiel sinnvoll? Was sind die Stärken, was sind die Schwächen?