Samuel Ferraz-Leite, Dirk Praetorius
Christoph Überhuber

Sommersemester 2008
11. Juni 2008

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

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 `serie11` *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 101*. Schreiben Sie eine Funktion `plotPotential(f)`, die den Plot eines Potentials $f = f(x, y)$ als farbige Projektion auf die Ebene erstellt. Zeichnen Sie 9 Konturlinien in den Plot. Geben Sie unter den Plot eine horizontale `colorbar`. Testen Sie Ihre Funktion mit dem Potential $f(x, y) = x \cdot e^{-x^2-y^2}$. Speichern Sie das entstandene Bild als farbige Post-Script-Datei unter `potential.eps` auf der `cad.zserv` ab. Das bild können Sie mit dem Unix-Befehl `gv potential.eps` betrachten.

Aufgabe 102*. Implementieren Sie das Gauss'sche Eliminationsverfahren aus Aufgabe 55 als Matlab-Funktion. Implementieren Sie die Funktion ein Mal direkt über Schleifen. Programmieren Sie danach eine verbesserte Version, in dem Sie wo sinnvoll Schleifen durch geeignete Matlab-Arithmetik ersetzen. indem Sie wo sinnvoll

Aufgabe 103*. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und Vektoren $I, A \in \mathbb{R}^N$ sowie $J \in \mathbb{R}^{n+1}$, die eine schwachbesetze $(m \times n)$-Matrix in CCS-Speicherung repräsentieren. Schreiben Sie eine Funktion `ccs2naive(I,J,A,m,n)`, die als Ergebnis die entsprechenden Vektoren bei naiver Speicherung zurückliefert.

Aufgabe 104*. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und 3 Vektoren $I, A \in \mathbb{R}^N$ und $J \in \mathbb{R}^{n+1}$, die eine schwachbesetzte Matrix in CCS-Speicherung repräsentieren, sowie ein Vektor $x \in \mathbb{R}^n$. Schreiben Sie mittels geeigneter (möglichst weniger) Schleifen eine Matrix-Vektor-Multiplikation `mvmsparse(I,J,A,m,n,x)`.

Aufgabe 105. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und 3 Vektoren $I, J, A \in \mathbb{R}^N$ mit $N \leq mn$, die eine schwachbesetzte Matrix in naiver Speicherung repräsentieren. Schreiben Sie eine Funktion `naive2ccs(I,J,A,m,n)`, die als Ergebnis die entsprechenden Vektoren des CCS-Formats zurückliefert.

Aufgabe 106. Verbessern Sie das Gauss'sche Eliminationsverfahren aus Aufgabe 103 wie in Aufgabe 56 beschrieben um eine Pivotstrategie. Schreiben Sie ein Skript, dass Ihre Funktion aus folgende Weise überprüft:

- Es soll eine reguläre 100×100 Matrix A aus Zufallszahlen erzeugt werden.
- Das Gauss'sche Eliminationsverfahren soll mit einer rechten Seite $b \in \mathbb{R}^{100}$ aus Zufallszahlen durchgeführt werden.

- Vergleichen Sie numerisch (also bis auf Rechenfehler der Größenordnung 10^{-12}) das Ergebnis der Matrix-Vektor-Multiplikation Ax mit der rechten Seite b, wobei $x \in \mathbb{R}^{100}$ die vom Eliminationsverfahren berechnete Lösung des Gleichungssystems $Ax = b$ ist.

Aufgabe 107. Zu gegebenen reellen Stützstellen $x_0 < x_1 < \cdots < x_n$ und Funktionswerten $y_j \in \mathbb{R}$ existiert ein eindeutiges Polynom $p(t)$ vom Grad n mit $p(x_j) = y_j$ für alle $j = 0, \ldots, n$. Es ist oft vorteilhaft, das Polynom $p(t)$ in der Newton-Basis darzustellen,

$$p(t) = a_0 + a_1(t - x_0) + a_2(t - x_0)(t - x_1) + \ldots + a_n(t - x_0) \cdots (t - x_{n-1}).$$

Die Koeffizienten $a_0, a_1, \ldots, a_n \in \mathbb{R}$ des Polynoms kann man mit dem *Verfahren der dividierten Differenzen* berechnen. Dazu definiert man für $j, m \in \mathbb{N}_0$ mit $j + m \leq n$ die Werte

$$y_{j,0} := y_j,$$
$$y_{j,m} := \frac{y_{j+1,m-1} - y_{j,m-1}}{x_{j+m} - x_j}.$$

Es gilt dann $a_\ell = y_{0,\ell}$. Es soll eine Funktion `newton` geschrieben werden, die bei Übergabe der Vektoren x und y den Koeffizientenvektor $a \in \mathbb{R}^{n+1}$ berechnet. Dazu berücksichtige man das Schema

$$\begin{array}{lllllllll}
y_0 & = & y_{0,0} & \longrightarrow & y_{0,1} & \longrightarrow & y_{0,2} & \longrightarrow \cdots \longrightarrow & y_{0,n} \\
 & & & \nearrow & & \nearrow & & \nearrow & \\
y_1 & = & y_{1,0} & \longrightarrow & y_{1,1} & & & \nearrow & \\
 & & & \nearrow & & & \nearrow & & \\
y_2 & = & y_{2,0} & \longrightarrow & \vdots & & & & \\
\vdots & & \vdots & & \vdots & \nearrow & & & \\
y_{n-1} & = & y_{n-1,0} & \longrightarrow & y_{n-1,1} & & & & \\
 & & & \nearrow & & & & & \\
y_n & = & y_{n,0} & & & & & &
\end{array}$$

Wenn möglich, schreibe man die Funktion so, dass *keine* Matrix $(y_{j,m})_{j,m=0}^{n}$ aufgebaut wird, sondern die y_j-Werte geeignet überschrieben werden.

Aufgabe 108. Man schreibe eine rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Dazu verwende man das Additionstheorem

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

Ferner schreibe man eine Funktion `binomial2`, die den Binomialkoeffizienten mittels

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

berechnet. Dazu ist eine rekursive Funktion `faktorielle` zu entwickeln, die zu gegebenem $n \in \mathbb{N}$ die Faktorielle $n!$ berechnet. Vergleichen Sie die Laufzeit der Implementierungen des Binomialkoeffizienten aus dieser Aufgabe miteinander und mit einer voll induktiven Implementierung wie in Aufgabe 14 beschrieben. Begründen Sie die beobachteten Effizienzunterschiede.

Aufgabe 109. Gegeben sei ein sortierter Vektor x der Länge n. Man schreibe eine Matlab-Funktion `findbisection(x,y)`, die einen Index i zurückgibt, für den $x_i = y$ gilt. Falls y nicht in x vorkommt soll 0 zurückgegeben werden. Naive Realisierung führt auf Aufwand $\mathcal{O}(n)$ im worst case, d.h. man durchsucht den Vektor von vorne nach hinten und das gesuchte y steht gerade an der letzten Stelle in x bzw. kommt überhaupt nicht in x vor. Wie kann man den Bisektionsalgorithmus geeignet modifizieren, um einen Algorithmus mit worst case Aufwand $\mathcal{O}(\log(n))$ zu erhalten?

Aufgabe 110. Gegeben seien $x, y \in \mathbb{N}$. Man schreibe eine rekursive Funktion `test`, die herausfindet, ob $x = y^n$ für ein geeignetes $n \in \mathbb{N}_0$ gilt. In diesem Fall werde $n \in \mathbb{N}_0$ zurückgegeben, anderenfalls -1.