Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2010/11
16. Dezember 2010

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1*. Nicht jede Matrix $A \in \mathbb{R}^{n\times n}$ hat eine normalisierte LU-Zerlegung $A = LU$, d.h.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \dots & 0 & u_{nn} \end{pmatrix}.$$

Wenn aber A eine normalisierte LU-Zerlegung besitzt, so gilt

$$u_{ik} = a_{ik} - \sum_{j=1}^{i-1} \ell_{ij} u_{jk} \quad \text{für } i = 1, \dots, n, \quad k = i, \dots, n,$$

$$\ell_{ki} = \frac{1}{u_{ii}} \Big(a_{ki} - \sum_{j=1}^{i-1} \ell_{kj} u_{ji} \Big) \quad \text{für } i = 1, \dots, n, \quad k = i+1, \dots, n,$$

$$\ell_{ii} = 1 \quad \text{für } i = 1, \dots, n,$$

wie man leicht über die Formel für die Matrix-Matrix-Multiplikation zeigen kann. Alle übrigen Einträge von $L, U \in \mathbb{R}^{n\times n}$ sind Null. Schreiben Sie eine Funktion `computeLU`, die die LU-Zerlegung von A berechnet und zurückgibt. Dazu überlege man, in welcher Reihenfolge man die Einträge von L und U berechnen muss, damit die angegebenen Formeln wohldefiniert sind (d.h. alles was benötigt wird, ist bereits zuvor berechnet worden). Speichern Sie den Source-Code unter `computeLU.m` in das Verzeichnis `serie09`.

Aufgabe 9.2*. Gegeben sei eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit $\ell_{jj} \neq 0$ für alle $j = 1, \dots, n$. Zu gegebenem $y \in \mathbb{R}^n$ existiert dann ein eindeutiges $x \in \mathbb{R}^n$ mit $Lx = y$. Schreiben Sie eine Funktion `solveL`, die x berechnet. — Um den Algorithmus herzuleiten, schreibe man das Matrix-Vektor-Produkt $y = Lx$ komponentenweise für y_j mit $j = 1, \dots, n$ als Summe hin. Man überlege, wie die spezielle Gestalt von L die Laufindizes der Summe vereinfacht und löse diese Gleichung nach x_j auf. Speichern Sie den Source-Code unter `solveL.m` in das Verzeichnis `serie09`.

Aufgabe 9.3*. Schreiben Sie eine Funktion `solve`, die die Lösung x des linearen Gleichungssystem $Ax = b$ berechnet. Gehen Sie dabei wie folgt vor: Berechnen Sie die LU-Zerlegung und lösen Sie für die Faktorisierung $LUx = b$ in zwei Schritten: (1) $Ly = b$, (2) $Ux = y$. Überlegen Sie, dass Sie dadurch tatsächlich die Lösung x von $Ax = b$ berechnen. Speichern Sie den Source-Code unter `solve.m` in das Verzeichnis `serie09`.

Aufgabe 9.4*. 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 Rechteckregel* berechnet man für gegebenes $n \in \mathbb{N}$ und $h := (b-a)/n$ zum Beispiel

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

Schreiben Sie eine Funktion `rechteckregel(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 $(I_1, \dots, I_n)$ der Approximationen zurück. Man teste die numerische Integration am Beispiel $f(x) = \exp(x)$ auf dem Intervall $[0, 10]$ und gebe abhängig von n den Fehler $|I - I_n|$ tabellarisch aus. Speichern Sie den Source-Code unter `rechteckregel.m` in das Verzeichnis `serie09`.

Aufgabe 9.5. Man kann den Speicheraufwand in Aufgabe 9.3 minimieren, indem man die Einträge der Matrix A geeignet durch die Einträge der Matrizen L und U überschreibt. Ferner kann man den Vektor b bei geeignetem Vorgehen, zunächst durch y und schließlich durch x überschreiben. Dadurch wird insgesamt kein zusätzlicher Speicher benötigt. Realisieren Sie dieses Vorgehen.

Aufgabe 9.6. Die Determinante einer Matrix $A \in \mathbb{R}^{n\times n}$ kann man über die normalisierte LU-Zerlegung aus Aufgabe 9.1 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.

Aufgabe 9.7. Die Matrix $A \in \mathbb{R}^{n\times n}$ sei eine Tridiagonalmatrix, d.h.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & \\ & a_{3,2} & a_{3,3} & \ddots & \\ & & \ddots & \ddots & a_{n-1,n} \\ & & & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

wobei alle anderen Einträge von A Null sind, und besitze eine LU-Zerlegung. Mit Hilfe der Formeln in Aufgabe 9.1 überlege man sich Formeln für die Einträge von L und U in diesem speziellen Fall. Dann schreibe man eine Funktion

```
[L,U] = computeLU3(A)
```

bei der keine unnötigen Rechenoperationen (d.h. Additionen/Multiplikationen von Null) durchgeführt werden und nur Einträge von L und U berechnet werden, die nicht Null sind.

Aufgabe 9.8. 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 $p(a + jh) = f(a + jh)$. Schreiben Sie 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 $(I_1, \dots, I_n)$ 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 dem Fehler $|I - I_n|$ auch die experimentelle Konvergenzordnung tabellarisch aus.

Aufgabe 9.9. Schreiben Sie eine Funktion `merge`, die zwei aufsteigend sortierte Felder $a \in \mathbb{R}^m$ und $b \in \mathbb{R}^n$ so vereinigt, dass das resultierende Feld $c \in \mathbb{R}^{m+n}$ ebenfalls aufsteigend sortiert ist, z.B. soll $a = (1, 3, 3, 4, 7)$ und $b = (1, 2, 3, 8)$ als Ergebnis $c = (1, 1, 2, 3, 3, 3, 4, 7, 8)$ liefern. Dabei soll ausgenutzt werden, dass die Felder a und b bereits sortiert sind. Es soll insbesondere *kein* Sortierverfahren auf das Vereinigungsfeld [a,b] angewandt werden.

Aufgabe 9.10. Schreiben Sie eine rekursive Funktion `mergesort`, die ein Feld a aufsteigend sortiert und das sortierte Feld zurückgibt. Gehen Sie dabei nach folgender Strategie vor:

- Hat a Länge ≤ 2, so wird das Feld a explizit sortiert.
- Hat a Länge > 2, halbiert man a in zwei Teilfelder b und c. Man ruft rekursiv `mergesort` für b und c auf und vereinige die sortierten Teilfelder mittels `merge` aus Aufgabe 9.9.

Machen Sie sich das Vorgehen anhand des einfachen Beispiels $a = [1, 3, 5, 2, 7, 1, 1, 3]$ klar.