Dirk Praetorius,
Marcus Page

Sommersemester 2012
23.04.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 7

Aufgabe 7.1*. (verschiedene Matrixtypen) Schreiben Sie eine Struktur `Matrix` zur Speicherung von quadratischen $n \times n$ `double` Matrizen, in der neben vollbesetzten Matrizen (Typ `'F'`) auch untere (Typ `'L'`) und obere (Typ `'U'`) Dreiecksmatrizen gespeichert werden können. Dabei bezeichnet man Matrizen

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & u_{33} & \dots & u_{3n} \\ & & & \ddots & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix} \qquad L = \begin{pmatrix} \ell_{11} & & & & \mathbf{0} \\ \ell_{21} & \ell_{22} & & & \\ \ell_{31} & \ell_{32} & \ell_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{pmatrix}$$

als obere bzw. untere Dreiecksmatrix. Mathematisch formuliert, gilt also $u_{jk} = 0$ für $j > k$ bzw. $\ell_{jk} = 0$ für $j < k$. Eine vollbesetzte Matrix werde im Fortran-Format spaltenweise als dynamischer Vektor der Länge $n \cdot n$ gespeichert. Dreiecksmatrizen sollen in einem Vektor der Länge $\sum_{j=1}^{n} j = n(n+1)/2$ gespeichert werden. Schreiben Sie die Funktionen, um mit dieser Struktur arbeiten zu können (`newMatrix`, `delMatrix`, `getMatrixDimension`, `getMatrixType`, `getMatrixEntry`, `setMatrixEntry`). Dabei hängen insbesondere die Funktionen `getMatrixEntry` und `setMatrixEntry` vom Matrixtyp (Dreiecksmatrix!) ab. Speichern Sie den Source-Code unter `matrix.c` in das Verzeichnis `serie07`.

Aufgabe 7.2*. (Transposition) Schreiben Sie eine Funktion `transpose`, die zu einer dynamisch gespeicherten Matrix $A \in \mathbb{R}^{m \times n}$ (vom Typ `double*`) die transponierte Matrix $A^T \in \mathbb{R}^{n \times m}$ berechnet. Dabei sind die Einträge von A^T gerade durch $(A^T)_{jk} = A_{kj}$ definiert. Sowohl A, als auch A^T sollen in der Struktur aus Aufgabe 7.1 gespeichert werden. Überlegen Sie sich zunächst welche Form die transponierte Matrix A^T hat. Im Hauptprogramm sollen die Dimensionen $m, n \in \mathbb{N}$ sowie die Einträge der Matrix A eingelesen und A^T ausgegeben werden. Speichern Sie den Source-Code unter `transpose.c` in das Verzeichnis `serie07`.

Aufgabe 7.3*. (LU-Zerlegung) 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

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

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

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

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). Die Matrizen sollen in der Struktur aus Aufgabe 7.1 gespeichert werden. Speichern Sie den Source-Code unter `computeLU.c` in das Verzeichnis `serie07`.
Welchen Aufwand hat diese Berechnung? Geben Sie den Aufwand in $\mathcal{O}$-Notation an.

Aufgabe 7.4*. (LU-Zerlegung) Schreiben Sie eine Funktion `solveLU`, die die Lösung x des linearen Gleichungssystems $Ax = b$ berechnet. Dabei soll die folgende Lösungsstrategie verwendet werden:

(1) Berechne die LU-Zerlegung von A.

(2) Löse $Ly = b$ nach y.

(3) Löse $Ux = y$ nach x.

Es gilt dann nämlich $Ax = LUx = Ly = b$. Schleifen. Speichern Sie den Source-Code unter `solveLU.c` in das Verzeichnis `serie07`.

Aufgabe 7.5. (LU-Zerlegung) Man kann den Speicheraufwand in Aufgabe 7.4 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. Speichern Sie den Source-Code unter `solveLUefficient.c` in das Verzeichnis `serie07`.

Aufgabe 7.6. (LU-Zerlegung) Schreiben Sie eine Funktion `solve`, die die Lösung x des linearen Gleichungssystem $Ax = b$ berechnet. Für die Matrix verwende man die Form aus Aufgabe 7.1. Vektoren werden wie in der Vorlesung gespeichert. Dabei soll die Struktur des Gleichungssystems (Dreiecksystem!) ausgenutzt werden. Ist A keine Dreiecksmatrix, berechne man die LU-Zerlegung und löse die Faktorisierung $LUx = b$ in zwei Schritten: (1) $Ly = b$, (2) $Ux = y$. Speichern Sie den Source-Code unter `solve.c` in das Verzeichnis `serie07`.

Aufgabe 7.7. (LU-Zerlegung, Tridiagonalmatrizen) 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 7.3 überlege man sich Formeln für die Einträge von L und U in diesem speziellen Fall.

Dann schreibe man eine Funktion `computeLU3` 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. Speichern Sie den Source-Code unter `computeLU3.c` in das Verzeichnis `serie07`. Welchen Aufwand hat diese Berechnung? Geben Sie den Aufwand in $\mathcal{O}$-Notation an.

Aufgabe 7.8. Was ist der Unterschied zwischen einem Array und einer Struktur? Wann verwendet man was und warum?

Aufgabe 7.9. (ASCII-File schreiben) Schreiben Sie eine Funktion `saveMatrix`, die eine Matrix mit der Struktur aus Aufgabe 7.1 als vollbesetzte Matrix in einer ASCII Datei abspeichert. Etwaige Dreiecksmatrizen sollen hierbei also mit Nullen aufgefüllt werden. Speichern Sie den Source-Code unter `savematrix.c` in das Verzeichnis `serie07`.

Aufgabe 7.10. (ASCII-File auslesen) Schreiben Sie eine Funktion `saveMatrix`, die eine in einer ASCII Datei gespeicherte Matrix einliest, und in der Struktur aus Aufgabe 7.1 abspeichert. Der Typ der Matrix soll hierbei automatisch erkannt und entsprechend zugewiesen werden. Zum testen können Sie die Funktion `savematrix` aus der letzten Aufgabe verwenden. Speichern Sie den Source-Code unter `loadMatrix.c` in das Verzeichnis `serie07`.