Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
26.06.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 13

Hinweis. Wir betrachten die Klasse `Matrix` aus der Vorlesung (Folie 357) und die davon abgeleitete Klasse `SquareMatrix` (Folie 366 bzw. Aufgabe 12.3). Im Nachhinein wurden die Klassen `LowerTriangularMatrix` (Folie 373) und `DiagonalMatrix` (Aufgabe 12.6) von der Klasse `SquareMatrix` abgeleitet.

Aufgabe 13.1. 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

$$\begin{aligned} u_{ik} &= a_{ik} - \sum_{j=1}^{i-1} \ell_{ij} u_{jk} \quad \text{für } i = 1, \dots, n, \quad k = i, \dots, n, \\ \ell_{ki} &= \frac{1}{u_{ii}} \Big(a_{ki} - \sum_{j=1}^{i-1} \ell_{kj} u_{ji} \Big) \quad \text{für } i = 1, \dots, n, \quad k = i+1, \dots, n, \\ \ell_{ii} &= 1 \quad \text{für } i = 1, \dots, n. \end{aligned}$$

Leiten Sie diese Formeln aus der Formel für die Matrix-Matrix-Multiplikation her. Implementieren Sie für die Klasse `SquareMatrix` die Methode `computeLU`, welche die normalisierte LU-Zerlegung einer quadratischen Matrix $A \in \mathbb{R}^{n\times n}$ berechnet und zurückgibt. Der Rückgabewert $R \in \mathbb{R}^{n\times n}$ sei dabei wieder vom Typ `SquareMatrix`, wobei die beiden Dreiecksmatrizen L und U in R gespeichert werden sollen

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

Die Diagonale von L muss hierbei nicht explizit gespeichert werden (Warum?). Testen Sie Ihren Code entsprechend!

Aufgabe 13.2. Welchen Aufwand besitzt Ihre Implementierung der LU-Zerlegung aus Aufgabe 13.1? Schreiben Sie das Ergebnis in der $\mathcal{O}$-Notation auf und erklären Sie wie sie auf das Ergebnis gekommen sind.

Aufgabe 13.3. Die Determinante einer Matrix $A \in R^{n\times n}$ kann über die normalisierte LU-Zerlegung aus Aufgabe 13.1 berechnet werden: Es gilt nämlich $\det(A) = \det(L)\det(U) = \det(U) = \prod_{j=1}^{n} u_{jj}$. Erweitern Sie die Klasse `SquareMatrix` um eine Methode `det`, welche die Determinante über die LU-Zerlegung berechnet und zurückgibt. Die Matrix selbst soll hierbei nicht überschrieben werden. Testen Sie Ihren Code entsprechend!

Aufgabe 13.4. Erweitern Sie die Klasse `SquareMatrix` um eine Methode `solve`, welche die Lösung eines Gleichungssystems der Form $Ax = b$ mit Hilfe der LU-Zerlegung folgendermaßen berechnet. Für die Matrix $A = LU$ löst man zuerst $Ly = b$ und anschließend $Ux = y$. Testen Sie Ihren Code entsprechend!

Aufgabe 13.5. Eine schiefsymmetrische Matrix ist eine Matrix $S \in \mathbb{R}^{n\times n}$, die gleich dem Negativen ihrer Transponierten ist: $S^T = -S$. Leiten Sie von der Klasse `SquareMatrix` die Klasse `SkewSymmetricMatrix` ab. Zur Speicherung der Matrixeinträge verwenden Sie einen Vektor mit Länge $n(n-1)/2$. Warum ist das ausreichend? Implementieren Sie Konstruktoren, Type-Cast und den Koeffizientenzugriff. Für die Diagonaleinträge gehen Sie vor wie in der Klasse `LowerTriangularMatrix` aus der Vorlesung: Speichern Sie `double zero` und `double const_zero` auf die sie beim Koeffizientenzugriff dann zugreifen. Testen Sie Ihren Code entsprechend!

Aufgabe 13.6. Schreiben Sie eine Template-Funktion `pot(T x, int n)`, welche für einen beliebigen Datentyp `T` (der die Funktionen `operator*` und `operator/` unterstützt) die Funktion x^n realisiert. Testen Sie ihr Beispiel mit verschiedenen Datentypen. Speichern Sie den Source-Code unter `pot.cpp` in das Verzeichnis `serie13`.
Hinweis: Beachten Sie, dass die Funktion auch für negative n zu programmieren ist. Dafür können Sie verwenden, dass `x/x` dem Einselement von `operator*` vom Typ `T` entspricht.

Aufgabe 13.7. Erklären Sie den Unterschied zwischen `public`-, `private`-, und `protected`-Vererbung anhand eines selbst gewählten Beispieles.

Aufgabe 13.8. Was ist ein Gleitkommazahlsystem? Aus welchen Bestandteilen setzt sich eine Gleitkommazahl zusammen? Wie bestimmt man daraus ihren Wert? Was verbirgt sich hinter den Symbolen `Inf`, `-Inf` und `NaN`? Was ist eine normalisierte Gleitkommazahl? Was ist ein implizites erstes Bit? Welchen Wert haben die größte und die kleinste positive normalisierte Gleitkommazahl im `float`-Gleitkommazahlsystem $\mathbb{F}(2, 24, -126, 127)$?