Prof. J. Zech
B. Huber
J. Westermann

Wintersemester 2023/24

Einführung in die Numerik

Universität
Heidelberg

Aufgabenblatt 3

Aufgabe 3.1 Programmieraufgabe: LR-Zerlegung mit Pivotisierung [1+1+1+1+1=5 P]

In der Vorlesung haben wir die $\mathbf{LR}$-Zerlegung kennengelernt, um lineare Gleichungssysteme zu lösen. Dabei wird eine reguläre Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ in ein Produkt

$$\mathbf{A} = \mathbf{LR}$$

aus einer Linksdreiecksmatrix $\mathbf{L}$ und einer Rechtsdreiecksmatrix $\mathbf{R}$ zerlegt.

(3.1a) Liegt die Zerlegung $\mathbf{A} = \mathbf{LR}$ vor, so kann durch *Vorwärtssubstitution* gefolgt von *Rückwärtssubstitution* das lineare Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ effizient für verschiedene Werte von $\mathbf{b}$ gelöst werden.

Implementieren Sie in PYTHON:

(i) Eine Funktion `forward_substitution(L,b)`, die die Lösung $\mathbf{y}$ des Gleichungssystems $\mathbf{Ly} = \mathbf{b}$ zurückgibt, wobei $\mathbf{L} \in \mathbb{R}^{n\times n}$ eine Linksdreiecksmatrix mit Einsen auf der Hauptdiagonale (d.h. $\mathbf{L}_{ii} = 1 \; \forall i = 1, \ldots, n$) und $\mathbf{b} \in \mathbb{R}^n$ sind. Der Code soll nur auf die strikt unterhalb der Hauptdiagonale liegenden Elemente von $\mathbf{L}$ zugreifen.

(ii) Eine Funktion `backward_substitution(R,b)`, die die Lösung $\mathbf{x}$ des Gleichungssystems $\mathbf{Rx} = \mathbf{y}$ zurückgibt, wobei $\mathbf{R} \in \mathbb{R}^{n\times n}$ eine reguläre Rechtsdreiecksmatrix und $\mathbf{y} \in \mathbb{R}^n$ sind. Der Code soll nur auf die auf oder oberhalb der Hauptdiagonale liegenden Elemente von $\mathbf{R}$ zugreifen.

(3.1b) In der Vorlesung haben wir gesehen, dass unter bestimmten Vorraussetzungen die $\mathbf{LR}$-Zerlegung durch *Gauss-Elimination ohne Pivotisierung* berechnet werden kann.

Implementieren Sie in PYTHON eine Funktion `LR(A)`, die für eine Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ die $\mathbf{LR}$-Zerlegung ohne Pivotisierung berechnet. Dabei sollte die Berechnung *in-place* stattfinden, also die Faktoren $\mathbf{L}$ und $\mathbf{R}$ direkt im Speicher von $\mathbf{A}$ konstruiert werden.

Testen Sie Ihre Implementation unter Verwendung von Teilaufgabe (3.1a).

(3.1c) Die Gauss-Elimination ohne Pivotisierung scheitert, wenn ein Pivotelement $a_{kk}^{(k-1)} = 0$ auftritt. Zudem werden wir sehen, dass der Algorithmus numerisch instabil ist, wenn für ein oder mehrere Pivotelemente gilt $|a_{kk}^{(k-1)}| \ll 1$.

Dies kann durch *Spaltenpivotisierung* verhindert werden. Dabei bestimmen wir im k-ten Schritt der Gauss-Elimination den Index $i = \mathrm{argmax}_{j\geq k} |a_{jk}^{(k-1)}|$ und vertauschen dann die Zeilen i und k von $\mathbf{A}^{k-1}$. Der Rest des Algorithmus bleibt unverändert.

Dadurch ergibt sich eine Zerlegung der Form

$$\mathbf{PA} = \mathbf{LR}$$

wobei $\mathbf{P}$ eine *Permutationsmatrix* (vgl. Definition 2.25) ist, die die Zeilenvertauschungen beschreibt. Die Lösung des Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ kann nun bestimmt werden, indem man zunächst die Zeilenvertauschungen auf $\mathbf{b}$ anwendet, also $\mathbf{b}' = \mathbf{Pb}$ bestimmt und dann $\mathbf{Ly} = \mathbf{b}', \mathbf{Rx} = \mathbf{y}$ durch Vorwärts- und Rückwärtseinsetzen löst.

Implementieren Sie in PYTHON eine Funktion `LR_partial_pivot(A)`, die für eine Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ die $\mathbf{LR}$-Zerlegung mit Spaltenpivotisierung berechnet. Dabei sollte die Berechnung *in-place* stattfinden, also die Faktoren $\mathbf{L}$ und $\mathbf{R}$ direkt im Speicher von $\mathbf{A}$ konstruiert werden. Weiterhin soll die Funktion einen Permutationsvektor $\mathbf{p} = (p_i)_{i=1}^n$ zurückgeben, der die Zeilenvertauschungen encodiert, d.h.

$$(\mathbf{Pa})_i = \mathbf{a}_{p_i} \qquad \forall \mathbf{a} \in \mathbb{R}^n$$

Testen Sie Ihre Implementation unter Verwendung von Teilaufgabe (3.1a).

(3.1d) In der Vorlesung werden wir sehen, dass *Spaltenpivotisierung* ausreichend ist, um einen in der Praxis numerischen stabilen Algrithmus zur Berechnung einer $\mathbf{LR}$-Zerlegung von regulären Matrizen zu erhalten.

Allerdings kann die Stabilität weiter verbessert werden, indem man eine *Vollpivotisierung* durchführt. Dabei bestimmen wir im k-ten Schritt $i_1, i_2 = \operatorname{argmax}_{j_1,j_2\geq k} |a_{j_1 j_2}^{(k-1)}|$ und vertauschen dann sowohl die Zeilen i_1 und k als auch die Spalten i_2 und k von $\mathbf{A}^{k-1}$.

Dadurch ergibt sich eine Zerlegung der Form

$$\mathbf{PAQ} = \mathbf{LR}$$

wobei $\mathbf{P}, \mathbf{Q}$ Permutationsmatrizen, die jeweils die Zeilen- und Spaltenvertauschungen beschreiben. Die Lösung des Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ kann nun bestimmt werden, indem man zunächst die Zeilenvertauschungen auf $\mathbf{b}$ anwendet, also $\mathbf{b}' = \mathbf{Pb}$ bestimmt, dann $\mathbf{Ly} = \mathbf{b}', \mathbf{Rx}' = \mathbf{y}$ durch Vorwärts- und Rückwärtseinsetzen löst und zuletzt noch $\mathbf{x} = \mathbf{Qx}'$ bestimmt.

Implementieren Sie in PYTHON eine Funktion `LR_full_pivot(A)`, die für eine Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ die $\mathbf{LR}$-Zerlegung mit Spaltenpivotisierung berechnet. Dabei sollte die Berechnung *in-place* stattfinden, also die Faktoren $\mathbf{L}$ und $\mathbf{R}$ direkt im Speicher von $\mathbf{A}$ konstruiert werden. Weiterhin soll die Funktion Permutationsvektoren $\mathbf{p} = (p_i)_{i=1}^n$, $\mathbf{q} = (q_i)_{i=1}^n$ zurückgeben, die die Zeilen- und Spaltenvertauschungen encodieren, d.h.

$$(\mathbf{Pa})_i = \mathbf{a}_{p_i}, \qquad (\mathbf{Qa})_i = \mathbf{a}_{q_i} \qquad \forall \mathbf{a} \in \mathbb{R}^n$$

Testen Sie Ihre Implementation unter Verwendung von Teilaufgabe (3.1a).

HINWEIS: Beachten Sie, dass die Vertauschungen der Einträge von $\mathbf{x}'$ in umgekehrter Reihenfolge zu den Spaltenvertauschungen in $\mathbf{A}$ erfolgen müssen.

(3.1e) Bestimmen Sie die Zeit zur Berechnung der $\mathbf{LR}$-Zerlegung ohne Pivotisierung, mit Spaltenpivotisierung und mit Vollpivotisierung für $\mathbf{A} \in \mathbb{R}^{n\times n}$ für verschieden große $n \in \mathbb{N}$.

Stellen Sie Ihre Ergebnisse graphisch dar und kommentieren Sie diese.

Aufgabe 3.2 LR-Zerlegung von strikt diagonaldominanten Matrizen [1+0.5+1.5+1=4 P]

Wie wir in der Vorlesung sehen werden, erlauben symmetrisch positiv definite Matrizen numerisch stabile Gauss-Elimination ohne Pivotisierung. In diesem Problem betrachten wir eine weitere wichtige Klasse von Matrizen, die diese angenehme Eigenschaft haben.

Definition. Eine Matrix $\mathbf{A} = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ heißt *strikt (spalten-)diagonaldominant*, wenn in jeder Spalte der Matrix der Absolutbetrag des Diagonaleintrags größer als die Summe der Absolutbeträge der anderen Einträge der Spalte sind, d.h.

$$|a_{jj}| > \sum_{\substack{i=1\\ j\neq i}}^{n} |a_{ij}| \quad \forall\, j = 1,\dots,n\ . \tag{3.2.1}$$

(3.2a) Zeigen Sie, dass jede strikt diagonaldominante Matrix regulär ist.

HINWEIS: Betrachten Sie ein Element des Kerns von $\mathbf{A}$ und eine betragsmäßig größte Komponente.

(3.2b) Zeigen Sie, dass der Gauss-Algorithmus mit Spaltenpivotsuche (siehe Aufgabe 1 oder auch Algorithmus 2.27 im Skript) angewandt auf eine strikt diagonal dominante Matrix im ersten Schritt stets a_{11} als Pivot wählt.

(3.2c) Sei $\mathbf{A}^{(1)}$ die Matrix, die sich nach dem ersten Schritt des Gauss-Algorithmus angewandt auf eine strikt diagonaldominante Matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ ergibt, d.h. nach den Zeilentransformationen, die die Einträge unter dem ersten Diagonaleintrag verschwinden lassen.

Zeigen Sie, dass die Submatrix $\mathbf{A}^{(1)}[1:, 1:]$ (Indizierung wie in PYTHON) strikt diagonaldominant ist.

HINWEIS: Stellen Sie zunächst eine Formel für die Einträge von $\mathbf{A}^{(1)}$ auf.

(3.2d) Zeigen Sie, dass bei der Anwendung des Gauss-Algorithmus mit Spaltenpivotsuche auf eine strikt diagonaldominante Matrix keine Zeilenvertauschung auftritt.

Aufgabe 3.3 Matrix 2-Norm [1+1+1=3 P]

(3.3a) Sei $\mathbf{A} \in \mathbb{R}^{m\times n}$ gegeben. Zeigen Sie, dass $\mathbf{x} \in \mathbb{R}^n$ existiert mit $\|\mathbf{x}\|_2 = 1$, so dass $\mathbf{A}^\top\mathbf{A}\mathbf{x} = \|\mathbf{A}\|_2^2\mathbf{x}$.

(3.3b) Zeigen Sie unter Verwendung von (3.3a), dass $\|\mathbf{A}\|_2^2 \leq \|\mathbf{A}\|_1\|\mathbf{A}\|_\infty$ für alle $\mathbf{A} \in \mathbb{R}^{m\times n}$.

(3.3c) Die *Konditionszahl* einer regulären Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ bzgl. einer Matrixnorm $\|\cdot\|$ ist definiert als

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Zeigen Sie folgende Resultate für die Konditionszahlen $\kappa_1(\mathbf{A})$, $\kappa_2(\mathbf{A})$ und $\kappa_\infty(\mathbf{A})$ einer regulären Matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ bzgl. der Matrixnormen $\|\cdot\|_1$, $\|\cdot\|_2$ bzw. $\|\cdot\|_\infty$:

(i) $\kappa_2(\mathbf{A})^2 \leq \kappa_1(\mathbf{A}) \cdot \kappa_\infty(\mathbf{A})$.

(ii) $\frac{1}{n}\kappa_2(\mathbf{A}) \leq \kappa_1(\mathbf{A}) \leq n\kappa_2(\mathbf{A})$.

HINWEIS: Wiederholen Sie ggf. die Inhalte von Anhang A.7 des Skripts.

Aufgabe 3.4 Kondition von Multiplikation, Division und Subtraktion [1+1.5+0.5=3 P]

(3.4a) Wir betrachten zunächst die Abbildung "Subtraktion", gegeben durch $f_{sub} : V_i \to V_o$, $(a, b) \mapsto a - b$, für $V_i := \mathbb{R}^2$ und $V_o := \mathbb{R}$. Für $p \in [1, \infty]$ versehen wir $\mathbb{R}^2$ mit der $\|\cdot\|_p$-(Vektor)norm.

Zeigen Sie, dass für alle $a, b \in \mathbb{R}$ mit $a \neq b$ gilt

$$\operatorname{cond}(f_{sub}, (a, b)) = \frac{2^{1-\frac{1}{p}}}{|a-b|}(|a|^p + |b|^p)^{1/p} \qquad p < \infty \tag{3.4.1}$$

und

$$\operatorname{cond}(f_{sub}, (a, b)) = \frac{2}{|a-b|} \max\{|a|, |b|\} \qquad p = \infty \tag{3.4.2}$$

HINWEIS: Verwenden Sie, ohne Beweis, dass für die p-Matrix(!)norm gilt $\left\|\begin{pmatrix}1 & -1\end{pmatrix}\right\|_p = 2^{1-\frac{1}{p}}$.

(3.4b) Wir betrachten die Abbildungen "Multiplikation" gegeben durch $f_{mul} : V_i \to V_o$, $(a, b) \mapsto a \cdot b$ und "Division" gegeben durch $f_{div} : V_i \backslash (\mathbb{R} \times \{0\}) \to V_o$, $(a, b) \mapsto \frac{a}{b}$. Wir versehen $\mathbb{R}^2$ mit der $\|\cdot\|_2$-(Vektor)norm.

Zeigen Sie, dass $\operatorname{cond}(f_{mul}, (a, b)) = \operatorname{cond}(f_{div}, (a, b))$ für alle $a, b \in \mathbb{R} \backslash \{0\}$.

(3.4c) Für welche Werte (a, b) sind f_{mul} und f_{div} gut bzw. schlecht konditioniert? Begründen Sie Ihre Antwort.

Veröffentlicht am 6. November 2023.

Abgabe bis zum 13. November 2023.

PYTHON: Geben Sie ein Jupyter-Notebook ab, dass allen geforderten Code sowie Ergebnisse enthält. Versehen Sie den Code sofern zum Verständnis nötig mit Kommentaren. Kommentieren Sie ggf. die Ergebnisse in Markdown-Zellen.