Prof. Dr. Dirk Lebiedz
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

Angewandte Numerik 1
SoSe 2020
08.05.2020

Lösungsvorschlag Übungsblatt 03

Besprechung in den Tutorien in der Woche vom 18.05.2020 bis 22.05.2020

Für dieses Übungsblatt gibt es 30 Theorie- und 12 Matlab-Punkte.
Die 70-Prozent-Grenzen liegen aktuell (inklusive Blatt 03) bei 62,3 Theorie- und 16,8 Matlabpunkten.

Aufgabe 7 *(Relative Konditionszahlen einer Matrix)* (7T+1T Punkte)

Für jeden festen Wert $c \in \mathbb{R}$ sei die Matrix A_c definiert durch $A_c := \begin{pmatrix} 1 & c \\ 0 & 1 \end{pmatrix}$.

a) Berechnen Sie die relativen Konditionszahlen $\kappa_{\|\cdot\|_1}(A_c)$, $\kappa_{\|\cdot\|_2}(A_c)$ und $\kappa_{\|\cdot\|_\infty}(A_c)$ der Matrix A_c in Abhängigkeit von c. Dabei stehen die Matrix-Normen $\|\cdot\|_1$, $\|\cdot\|_2$ und $\|\cdot\|_\infty$ für die aus der Vorlesung bekannten induzierten Matrixnormen.

b) Sei $\tilde{x}$ die Lösung des linearen Gleichungssystems $A_c\, x = b$ mit $b \in \mathbb{R}^2$. Das gestörte Gleichungssystem $A_c\, x = b + \Delta b$ habe die Lösung $(\tilde{x} + \Delta x)$. Außerdem wissen wir, dass $\|\Delta b\|_1/\|b\|_1 = 0.01$.

   Geben Sie eine obere Schranke für den relativen Fehler $\|\Delta x\|_1/\|\tilde{x}\|_1$ an.

Lösungsvorschlag:

a) Wir berechnen die relativen Konditionszahlen mit $\kappa(A) = ||A|| \cdot ||A^{-1}||$. Zum Berechnen von A^{-1} benutzen wir

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \Longrightarrow \quad B^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Somit erhalten wir:

$$A_c = \begin{pmatrix} 1 & c \\ 0 & 1 \end{pmatrix} \quad \Longrightarrow \quad A_c^{-1} = \begin{pmatrix} 1 & -c \\ 0 & 1 \end{pmatrix} = A_{-c}$$

Sowohl bei A_c, also auch bei A_{-c} sind die betragsmäßig größten Zeilen bzw. Spalten:

$$||A_c||_1 = ||A_c||_\infty = 1 + |c| = ||A_{-c}||_1 = ||A_{-c}||_\infty$$

Die relativen Konditionszahlen bezüglich $||\cdot||_1$ und $||\cdot||_\infty$ sind also:

$$\kappa_{||\cdot||_1}(A_c) = ||A_c||_1\, ||A_c^{-1}||_1 = ||A_c||_1\, ||A_{-c}||_1 = (1 + |c|)^2$$
$$\kappa_{||\cdot||_\infty}(A_c) = ||A_c||_\infty\, ||A_c^{-1}||_\infty = ||A_c||_\infty\, ||A_{-c}||_\infty = (1 + |c|)^2$$

Zur Berechnung der relativen Konditionszahl bezüglich der $||\cdot||_2$ benutzen wir, dass $||A||_2 = \sqrt{\lambda_{max}(A^TA)}$, wobei λ_{max} den betragsmäßig größten Eigenwert von A^TA bezeichnet.

$$\begin{aligned}
p_\lambda(A_c^TA_c) &= p_\lambda\begin{pmatrix}1 & c\\ c & c^2+1\end{pmatrix}\\
&= (1-\lambda)(c^2+1-\lambda)-c^2\\
&= c^2+1-\lambda-\lambda c^2-\lambda+\lambda^2-c^2\\
&= 1-\lambda(c^2+2)+\lambda^2 \overset{!}{=} 0 \quad\Longleftrightarrow\quad \lambda_{1,2} = \left(1+\frac{c^2}{2}\right)\pm\sqrt{\left(1+\frac{c^2}{2}\right)^2-1},\\
&\qquad \text{wobei} \quad \sqrt{\left(1+\frac{c^2}{2}\right)^2-1} = \sqrt{1+c^2+\frac{c^4}{4}-1} = \sqrt{c^2+\frac{c^4}{4}}
\end{aligned}$$

$$\Longrightarrow \quad \lambda_{max}(A_c^TA_c) = \left(1+\frac{c^2}{2}\right)+\sqrt{c^2+\frac{c^4}{4}} = \lambda_{max}(A_{-c}^TA_{-c})$$

$$\begin{aligned}
\Longrightarrow \quad \kappa_{||\cdot||_2}(A_c) &= ||A_c||_2||A_c^{-1}||_2 = ||A_c||_2||A_{-c}||_2 = \left(\sqrt{\lambda_{max}(A_c^TA_c)}\right)^2\\
&= \lambda_{max}(A_c^TA_c) = 1+\frac{c^2}{2}+\sqrt{c^2+\frac{c^4}{4}}
\end{aligned}$$

b) Beim gestörten Linearen Gleichungssystem $A_c x = b + \Delta b$ ist nur die rechte Seite gestört. Daher ergibt sich mit Satz 3.7 für den relativen Fehler der Lösung in der 1-Norm

$$\frac{||\Delta x||_1}{||\tilde{x}||_1} \le \kappa_{||\cdot||_1}(A_c)\frac{||\Delta b||_1}{||b||_1} = (1+|c|)^2\cdot 0.01 = \frac{(1+|c|)^2}{100}.$$

Aufgabe 8 *(Programmier-Aufgabe: Äquilibrierung einer Matrix)* (8M+2T+4M Punkte)

a) Schreiben Sie eine MATLAB-Funktion `D = normalizeMat(A, norm, type)`, die für eine reguläre Matrix $A \in \mathbb{R}^{n\times n}$ eine Äquilibrierung durchführt. Dabei ist

- `A` die zu äquilibrierende reguläre Matrix,
- `norm` ein String, der festlegt, ob bezüglich der Maximumnorm (`norm == 'infty'`) oder bezüglich der Summennorm (`norm == 'one'`) äquilibriert werden soll;
- `type` ein String, der die Art der Äquilibrierung festlegt. Es soll gelten

  $$\|(DA)_i\| = 1 \quad \forall\, i \in \{1,\ldots,n\}, \qquad \text{falls } \texttt{type == 'rows'} \quad \text{und}$$
  $$\|(AD)^{(j)}\| = 1 \quad \forall\, j \in \{1,\ldots,n\}, \qquad \text{falls } \texttt{type == 'cols'},$$

  wobei $(DA)_i$ die i-te Zeile der Matrix DA und $(AD)^{(j)}$ die j-te Spalte der Matrix AD bezeichnen;
- `D` die Diagonalmatrix, die die Matrix A äqulilibriert.

Hinweise: Die MATLAB-internen Funktionen zur Berechnung einer Norm dürfen in dieser Teilaufgabe nicht verwendet werden. Es ist möglich, diese Teilaufgabe ohne Verwendung einer Schleife zu lösen.

b) Geben Sie Optimalitätseigenschaften für die Matrizen DA und AD an.

c) Schreiben Sie ein Testskript `main8.m`, mit dem Sie Ihre in Aufgabenteil a) geschriebene Funktion `normalizeMat` für jede Kombination von zulässigen Eingaben für `norm` und `type` mindestens einmal anhand einer geeigneten Matrix testen.

Lösungsvorschlag:

a) Siehe Moodle-Kurs im Abschnitt „Lösungsvorschläge“ unter „Blatt 03 Aufgabe 8 Matlab-Lösung“.

b) Betrachten wir zunächst die Zeilenskalierung, also die Multplikation der Matrix A mit einer Diagonalmatrix von links:

Wir wählen entsprechend der Folie 62 aus dem Skript als Diagonalmatrix die Matrix D_z, deren Diagonaleinträge definiert sind durch

$$d_i = \left(\sum_{j=1}^{n} |a_{i,j}| \right)^{-1}.$$

Somit ist also d_i gerade das Inverse der (Betrags-)Summennorm der jeweilgen Zeile i der Matrix A. Damit gilt für jede Zeile i der skalierten Matrix $D_z A$

$$\sum_{j=1}^{n} |(D_z A)_{i,j}| = \sum_{j=1}^{n} |d_i a_{i,j}| = d_i \sum_{j=1}^{n} |a_{i,j}| = \left(\sum_{j=1}^{n} |a_{i,j}| \right)^{-1} \sum_{j=1}^{n} |a_{i,j}| = 1.$$

Nach Folie 69 ergibt sich für die Zeilenskalierung mit dieser Diagonalmatrix D_z die Optimalitätseigenschaft

$$\kappa_\infty(D_z A) \leq \kappa_\infty(DA) \quad \text{für jede reguläre Diagonalmatrix D.}$$

Durch Skalieren der Zeilen mit D_z wird also die Kondition der Matrix DA bzgl. der Zeilensummennorm optimiert.

Die Spaltenäquilibrierung entspricht der Multiplikation der Matrix A mit einer Diagonalmatrix D_s von rechts. Dabei werden alle Einträge der j-ten Spalte von A mit dem j-ten Diagonalelemement d_j der Diagonalmatrix D_s multipliziert.

Wenn wir für die Diagonaleinträge d_j gerade das Inverse der (Betrags-)Summennorm der jeweilgen Spalte j der Matrix A wählen, also

$$d_j = \left(\sum_{i=1}^{n} |a_{i,j}| \right)^{-1},$$

erhalten wir die Optimalitätseigenschaft

$$\kappa_1(AD_s) \leq \kappa_1(AD) \quad \text{für jede reguläre Diagonalmatrix D.}$$

Durch Skalieren der Spalten mit D_s wird also die Kondition der Matrix AD bzgl. der Spaltensummennorm optimiert.

Zur Spaltenäquilibrierung siehe beispielsweise `https://de.wikipedia.org/wiki/Äquilibrierung`, abgerufen am 02.08.2020, 17:50 Uhr.

c) Siehe Moodle-Kurs im Abschnitt „Lösungsvorschläge“ unter „Blatt 03 Aufgabe 8 Matlab-Lösung“.

Aufgabe 9 *(Aufwand zur Berechnung der Lösung eines linearen Gleichungssystems)* (2T+2T Punkte)

Der Aufwand zur Berechnung der Determinate $\det A$ einer Matrix $A \in \mathbb{R}^{n \times n}$ nach dem Laplace'schen Entwicklungssatz beträgt ungefähr $n!$ Operationen. Da zur Berechnung der Lösung x^* eines linearen Gleichungssystems $Ax = b$ mit $A \in \mathbb{R}^{n \times n}$ und $b \in \mathbb{R}^n$ mit Hilfe der Cramerschen Regel die Berechnung von $n+1$ Determinanten und n Quotienten notwendig sind, ergibt sich insgesamt ein Aufwand von $(n+1)! + n$ Operationen zur Berechnung der Lösung $x^* = A^{-1}b$ des linearen Gleichungssystems.

Im März 2020 wurde an der Universität Ulm der neue Hochleistungsrechner JUSTUS 2 eingeweiht. JUSTUS 2 hat eine theoretische Leistungsfähigkeit von 2 Petaflops und gehörte zum Zeitpunkt seiner Einweihung zu den 400 leistungsstärksten Supercomputern der Welt.
Wie lange würde dieser Supercomputer benötigen, um ein lineares Gleichungssystem der Dimension $n = 25$ mittels der Cramerschen Regel unter Verwendung des Laplace'schen Entwicklungssatzes zu lösen.
Hinweis: Ein Jahr hat rund $365,242$ Tage.

Lösungsvorschlag:

Mit Hilfe der Cramerschen Regel und des Laplace'schen Entwicklungssatzes würde der Computer folgende Zeit in Jahren benötigen:

$$\frac{(25+1)!+25}{2\cdot 10^{15}} \cdot \frac{1}{365.242 \cdot 24 \cdot 60 \cdot 60} \approx 6389 \text{ Jahre.}$$

Aufgabe 10 *(Aufwand zur Berechnung von Vektor- und Matrix-Multiplikationen)* (2T+2T+2T Punkte)
Seien im Folgenden $x, y \in \mathbb{R}^n$ zwei Spaltenvektoren und $A, B \in \mathbb{R}^{n\times n}$ zwei quadratische Matrizen.

a) Wie viele Additionen und wieviele Multiplikationen werden zur Berechnung des Vektor-Vektor-Produkts $x^T y$ benötigt. Geben Sie die Zahl der Operationen in $\mathcal{O}$-Notation an.

b) Geben Sie auch die Zahl der zur Berechnung des Matrix-Vektor-Produkts Ax benötigten Additionen und Multiplikationen an. Wie können Sie den benötigten Aufwand in $\mathcal{O}$-Notation angeben?

c) Wie viele Additionen und wieviele Multiplikationen benötigen Sie zur Berechnung des Matrix-Matrix-Produkts AB? Wie groß ist der Aufwand in $\mathcal{O}$-Notation?

Lösungsvorschlag:

a) $x^T y = \sum_{i=1}^n x_i \cdot y_i$, also $n-1$ Additionen und n Multiplikationen und somit ein Aufwand von $\mathcal{O}((n-1)+n) = \mathcal{O}(n)$.

Im Folgenden bezeichnet (c_i) die $i-te$ Zeile und (c_j) die $j-te$ Spalte einer Matrix C.

b) $Ax = (a_i)_{i=1,\ldots,n} \cdot x = \left(\sum_{j=1}^n a_{i,j} \cdot x_j\right)_{i=1,\ldots,n}$, also $n(n-1)$ Additionen und $n \cdot n$ Multiplikationen und somit ein Aufwand von $\mathcal{O}(n(n-1)+n\cdot n) = \mathcal{O}(2n^2-n) = \mathcal{O}(n^2)$.

b) $AB = (a_i \cdot B)_{i=1,\ldots,n} = (a_i \cdot b_j)_{i,j=1,\ldots,n} = (\sum_{k=1}^n a_{ik}\cdot b_{kj})_{i,j=1,\ldots,n}$, also $n \cdot n \cdot (n-1)$ Additionen und $n\cdot n\cdot n$ Multiplikationen und somit ein Aufwand von $\mathcal{O}(n\cdot n\cdot(n-1)+n\cdot n\cdot n) = \mathcal{O}(2n^3-n^2) = \mathcal{O}(n^3)$.

Aufgabe 11 *(LR-Zerlegung durch Koeffizientenvergleich)* (4T+3T+3T Punkte)
In dieser Aufgabe wollen wir zu einer gegebenen Matrix $A \in \mathbb{R}^{3\times 3}$ die LR-Zerlegung durch Koeffizientenvergleich bestimmen. Gesucht sind also eine linke untere normierte Dreiecksmatrix L und eine rechte obere Dreiecksmatrix R mit

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix} \quad \text{und} \quad R = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ 0 & r_{2,2} & r_{2,3} \\ 0 & 0 & r_{3,3} \end{pmatrix},$$

so dass $A = LR$.

a) Wie und in welcher Reihenfolge kann man die Koeffizienten $l_{i,j}$ und $r_{i,j}$ bestimmen.

b) Verallgemeinern Sie dies auf 4×4 –Matrizen und $n \times n$ –Matrizen.

c) Kann man für jede reguläre Matrix A eine LR-Zerlegung finden, so dass gilt $A = LR$? Begründen Sie Ihre Antwort.

Lösungsvorschlag:

a) Es gilt:

$$\begin{aligned}
LR &= \begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix} \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ 0 & r_{2,2} & r_{2,3} \\ 0 & 0 & r_{3,3} \end{pmatrix} \\
&= \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ l_{2,1} \cdot r_{1,1} & l_{2,1} \cdot r_{1,2} + r_{2,2} & l_{2,1} \cdot r_{1,3} + r_{2,3} \\ l_{3,1} \cdot r_{1,1} & l_{3,1} \cdot r_{1,2} + l_{3,2} \cdot r_{2,2} & l_{3,1} \cdot r_{1,3} + l_{3,2} \cdot r_{2,3} + r_{3,3} \end{pmatrix} \\
&\overset{!}{=} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix},
\end{aligned}$$

Die erste Spalte von L und R kann direkt berechnet werden, vorrausgesetzt $a_{1,1} \neq 0$:

$$\begin{aligned}
r_{1,1} &= a_{1,1} \\
a_{2,1} &= l_{2,1} \cdot r_{1,1} \iff l_{2,1} = \frac{a_{2,1}}{r_{1,1}} \\
a_{3,1} &= l_{3,1} \cdot r_{1,1} \iff l_{3,1} = \frac{a_{3,1}}{r_{1,1}}
\end{aligned}$$

Damit kann die zweite Spalte von L und R berechnet werden:

$$\begin{aligned}
r_{1,2} &= a_{1,2} \\
a_{2,2} &= l_{2,1} \cdot r_{1,2} + r_{2,2} \iff r_{2,2} = a_{2,2} - l_{2,1} \cdot r_{1,2} \\
a_{3,2} &= l_{3,1} \cdot r_{1,2} + l_{3,2} \cdot r_{2,2} \iff l_{3,2} = \frac{1}{r_{2,2}}(a_{3,2} - l_{3,1} \cdot r_{1,2})
\end{aligned}$$

Jetzt kann man alle noch fehlenden Einträge berechnen:

$$\begin{aligned}
r_{1,3} &= a_{1,3} \\
a_{2,3} &= l_{2,1} \cdot r_{1,3} + r_{2,3} \iff r_{2,3} = a_{2,3} - l_{2,1} \cdot r_{1,3} \\
a_{3,3} &= l_{3,1} \cdot r_{1,3} + l_{3,2} \cdot r_{2,3} + r_{3,3} \iff r_{3,3} = a_{3,3} - l_{3,1} \cdot r_{1,3} - l_{3,2} \cdot r_{2,3}
\end{aligned}$$

Die Einträge werden also spaltenweise berechnet. Es ist allerdings auch möglich die Einträge zeilenweise zu bestimmen.

b) Auch bei 4×4 und $n \times n$ - Matrizen können die Einträge von L und R spaltenweise durch Koeffizientenvergleich berechnet werden.

Es ergibt sich zum Beispiel folgender Algorithmus:

Vergleiche spaltenweise die Einträge von A und LR. In der j-ten Spalte:

- Berechne für $i = 1, \ldots, j$: $r_{i,j} = a_{i,j} - \sum\limits_{k=1}^{i-1} l_{i,k} \cdot r_{k,j}$
- Berechne für $i = j+1, \ldots, n$: $l_{i,j} = \frac{1}{r_{j,j}} \cdot \left(a_{i,j} - \sum\limits_{k=1}^{j-1} l_{i,k} \cdot r_{k,j} \right)$

c) Nein, man kann nicht für jede reguläre Matrix A eine LR-Zerlegung finden.
Betrachte zum Beispiel die reguläre 2×2 Matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Angenommen, es gäbe eine LR-Zerlegung von A. Dann würde gelten:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_{2,1} & 1 \end{pmatrix} \cdot \begin{pmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{pmatrix} = \begin{pmatrix} r_{1,1} & r_{1,2} \\ l_{2,1} \cdot r_{1,1} & l_{2,1} \cdot r_{1,2} + r_{2,2} \end{pmatrix}.$$

Also wäre $r_{1,1} = 0$ und damit $1 = l_{2,1} \cdot r_{1,1} = l_{2,1} \cdot 0 = 0$.

Das ist ein Widerspruch.

Daher gibt es für die reguläre Matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ keine LR-Zerlegung.

Hinweise:
Die Lösungen der Theorieaufgaben und Ihren Text zu den Programmieraufgaben können Sie in LaTeX erstellen oder handschriftlich aufschreiben und einscannen. Die Programmieraufgaben sind in Matlab zu lösen. Der Source Code muss strukturiert und dokumentiert sein.
Falls Sie die Aufgaben im Team lösen, geben Sie bitte auf allen Lösungen alle an der Aufgabe beteiligten Teammitglieder an. Jedes Teammitglied muss eine Lösung abgeben.
Speichern Sie Ihre Lösungen und Ihre Ergebnisse in einem Directory mit dem Namen **Blatt03_*Vorname*_*Nachname*** und verpacken Sie dieses in eine `.zip`-Datei. Laden Sie **spätestens 48 Stunden vor Ihrem Tutorium** diese `.zip`-Datei in Moodle hoch.