Institut für Analysis und Numerik
Universität Magdeburg
Thomas Richter
Gozel Judakova

22.05.2020

thomas.richter@ovgu.de
gozel.judakova@ovgu.de

Übung Nr. 6 zur Vorlesung Algorithmische Mathematik II
Sommersemester 2020

Abgabe bis zum 29. Mai 2020

Störungstheorie Es geht um die Fehleranfälligkeit beim Lösen von von linearen Gleichungssystemen $Ax = b$: Angenommen die rechte Seite b und / oder die Matrix A sind nicht exakt bekannt, sondern nur fehlerbehaftet, also $\tilde{A}\tilde{x} = \tilde{b}$ mit $\tilde{A} = A + \delta A$ und $\tilde{b} = b + \delta b$. Wie groß ist dann der Fehler in der Lösung $\tilde{x} - x$? In der praktischen Aufgabe zu Blatt 5 hat sich gezeigt, dass Python (mit dem Gauss'schen Eliminationsverfahren) nicht in der Lage ist, die Gleichungssysteme fehlerfrei zu lösen. Obwohl Matrix und rechte Seite höchstens mit dem Rundungsfehler in Größe der Maschinengenauigkeit versehen sind, war der Fehler im Ergebnis wesentlich.
Dies ist Them im ersten Abschnitt *7.2 Störungstheorie und Stabilitätsanalyse von linearen Gleichungssystemen*. Der zentrale Begriff ist die *Konditionszahl einer Matrix* die beschreibt, mit welcher Fehlerverstärkung zu rechnen ist. Die wichtigen Ergebnisse sind Satz 7.20 und 7.23 sowie Satz 7.24, der diese beiden Teilergebnisse zusammenfasst.
Bei der Herleitung der Fehlerabschätzungen wird intensiv Gebrauch von den Zusammenhängen zwischen üblicher Vektornorm und verträglicher Matrixnorm gemacht.
Im Abschnitt wird wieder häufig auf Eigenwerte zurückgegriffen. Die entsprechenden Teile sind jedoch für das weitere Vorgehen nicht wesentlich und können schnell übersprungen werden.
Man lese den Abschnitt und gebe kurze Antworten:

- Was ist die Konditionszahl einer Matrix?
- Was ist die Konditionszahl der Matrix

  $$A = \begin{pmatrix} 2 & 0 \\ 0 & -4 \end{pmatrix}$$

  gemessen in der $\|\cdot\|_\infty$-Norm?
- Mit welcher relativen Fehler im Ergebnis, d.h. $\|\delta x\|/\|x\|$ ist zu rechnen, wenn bei dieser Matrix die rechte Seite mit eine Fehler von 5% versehen ist? (Die Matrix sei fehlerfrei).

Von der Gauss-Elimination zur LR-Zerlegung Die Gauss-Elimination zum Lösen eines LGS sollte bereits bekannt sein. In der praktischen Anwendung geht man anders vor: Zunächst wird nur die Matrix A, nicht aber die rechte seite b bearbeitet. In einem ersten Schritt wird die Matrix A auf mit Gauss-Elimination auf Dreiecks-Gestalt gebracht (vergleiche Abschnitt 7.3 ganz oben). Dabei merken wir uns jedoch die einzelnen Transformationsschritte und erreichen so eine *multiplikative Zerlegung* der Matrix A in der Form

$$A = LR,$$

wobei L eine *linke unterer Dreiecksmatrix* ist (d.h. rechts oberhalb der Diagonale sind nur Nullen) und R eine *rechte obere Dreiecksmatrix* ist (links unterhalb der Diagonale nur Nullen). Im Abschluss wird das lineare Gleichungssystem in 2 Schritten gelöst

$$Ax = b \quad \Leftrightarrow \quad L\underbrace{Rx}_{=y} = b \quad \Leftrightarrow \quad \begin{matrix} Ly = b \\ Rx = y \end{matrix}$$

Insgesamt sind es also 3 Schritte: Matrix zerlegen $A = LR$, $Ly = b$ lösen (das sogenannte Vorwärtseinsetzen) und $Rx = b$ lösen (rückwärtseinsetzen). Der erste Schritt ist sehr aufwendig, das Lösen im Anschluss ist einfach. Der Vorteil dieser Zerlegung ist einerseits die Möglichkeit nach einmaliger Zerlegung der Matrix das LGS für verschiedene rechte Seiten zu lösen. Darüber hinaus kann die Zerlegung stabiler durchgeführt werden.
Man lese Abschnitt *7.3. Das Gauss'sche Eliminationsverfahren und die LR-Zerlegung* bis vor Satz 7.27. Wir kommen in der folgenden Woche noch einmal auf diesen Teil zurück. Wichtig ist insbesondere die Konstruktionsidee der schrittweisen Zerlegung der Matrix: Jeder Eliminationsschritt der Gauss-Elimination wird durch die Multiplikation mit einer Frobenius-Matrix erreicht. Schließlich ist:

$$R := F^{(n-1)}F^{(n-2)} \cdots F^{(1)}A$$

die rechte obere Dreiecksmatrix und die Produkte der Matrizen F bilden das Inverse der linken unteren Dreiecksmatrix.
Man gebe kurze Antworten

1. Was ist der Aufwand zum Lösen eines LGS mit einer Dreiecksmatrix, egal ob linke untere oder rechte obere?
2. Was ist die Ausgabe von

```
1 for i in reversed(range(2,4)):
2     print (i)
```

3. Was ist der Aufwand zur Berechnung der euklidischen Norm eines Vektors $\mathbb{R}^n$

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

4. Man beschreibe eine Frobenius-Matrix

Aufgabe 6.1:

Man beweise *Satz 7.26* (*Frobenius-Matrix*).

Hinweis: Beweisen ist ein großes Wort. Es geht hier um das Nachrechnen. Man schreibe die zu untersuchenden Produkte komponentenweise, d.h. z.B. $(AB)_{ij} = \sum_k A_{ik} B_{kj}$ *und nutze dann jeweils das Wissen über die Matrizen aus, d.h. das viele Einträge entweder Null oder 1 sind.*

Programmieraufgabe 6.2:

Wir betrachten die Gauss-Elimination gemäß

```
1  def gauss_elimination(A,b):
2      n = len(A)    #
3      assert len(b)==n,'A und b passen nicht!'
4
5      for i in range(n-1):
6          for j in range(i+1,n):
7              s=A[j,i]/A[i,i]
8              for k in range(i+1,n):
9                  A[j,k]=A[j,k]-s*A[i,k]
10             b[j]=b[j]-s*b[i]
11             A[j,i]=0
12     for i in reversed(range(n)):
13         b[i]=b[i]/A[i,i]
14         A[i,i]=1.0
15         for j in range(i):
16             b[j]=b[j]-b[i]*A[j,i]
17             A[j,i]=0.0
18     return b
```

a) Man verwende den numpy-slice Operator zur Beschleunigung von Zeilen 8-9 sowie 15-17. Für die angegebenen Testmatrix stelle man die Laufzeit beider Methoden (ohne und mit Beschleunigung) doppelt-logarithmisch graphisch dar. Hierbei sollen beide Zeitreihen in einem plot gezeigt werden.

b) Man erstelle eine Funktion zum Rückwärtseinsetzen. Als Testmatrix verwende man einfach die Matrizen aus Aufgabe **a**) und setze alle Einträge links unterhalb der Diagonalen auf Null. Man vergleiche wieder eine einfache und eine mit numpy-slices beschleunigte Variante. Hier kommt es nicht auf die Lösung eines LGS an. D.h., wir können die Rückwärtselimination zu einem beliebigem Vektor bei beliebiger Matrix durchführen. Es geht nur um die Messung des Aufwandes.

c) Man stelle die erreichte Beschleunigung in für beide Algorithmen, Gauss-Elimination und Rückwärtseinsetzen einfach-logarithmisch dar. Hierbei soll die x-Achse logarithmisch gewählt werden.

Hinweis: alles hierfür Notwendige ist Teil des auf der Homepage verlinkten Python-Kurses in den Abschnitten `vectors.ipynb` *sowie* `plot.ipnyb`*. Darüber hinaus finden Sie Algorithmen zum Rückwärtseinsetzen und auch Tipps zur Beschleunigung mit slices im Skript.*