Prof. Dr. Reinhard Klein
Alexander Schier

Universität Bonn
Institut für Informatik II
Wintersemester 2023/2024 20.12.2023

Übungen zu Angewandte Mathematik: Numerik - Blatt 11

Die Lösungen für die praktischen Aufgaben müssen bis Mittwoch, den 10.01.2024, um 12:00 im eCampus hochgeladen werden. Die Lösungen zu Theorieaufgaben müssen bis 12:00 in die Postfächer im Raum 0.004 im Hörsaalgebäude eingeworfen oder digital im eCampus abgegeben werden. Bei digitaler Abgabe werden werden keine Scans, Fotos, etc. gewertet. Weihnachtsaufgaben geben Bonuspunkte.

Aufgabe 1 (Matrixfunktionen, *4 Punkte*)

Viele wichtige Funktionen $f : \mathbb{C} \to \mathbb{C}$ lassen sich als Potenzreihe approximieren (z.B. $\exp, \sin, \cos$, Polynome und auf kleineren Definitionsbereichen auch $\log, \arccos, \sqrt{}$ und rationale Funktionen), d.h.

$$f(x) \approx \sum_{i=0}^{n} c_i \cdot x^i$$

mit Koeffizienten $c_0, c_1, \ldots, c_n \in \mathbb{C}$. Wir betrachten die Verallgemeinerung der Funktion f auf Matrizen $B \in \mathbb{C}^{m \times m}$:

$$g(B) := \sum_{i=0}^{n} c_i \cdot B^i \in \mathbb{C}^{m \times m}$$

Sei $A \in \mathbb{C}^{m \times m}$ eine quadratische Matrix mit Eigenwertzerlegung $A = V \cdot \Lambda \cdot V^{-1}$ wobei $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ die Diagonalmatrix der Eigenwerte ist und für jeden Eigenwert λ_i gelte $f(\lambda_i) < \infty$. Zeige, dass

$$g(A) = V \cdot \mathrm{diag}(f(\lambda_1), \ldots, f(\lambda_m)) \cdot V^{-1}.$$

Hinweis: Zeige zuerst per Induktion, dass $A^i = V \cdot \Lambda^i \cdot V^{-1}$ und verwende dies um die Aufgabe zu lösen.

Aufgabe 2 (Alle Eigenwerte berechnen, *3 Punkte*)

Implementiere das QR-Verfahren ohne Shifts (Algorithmus 7.3 im Skript) im gegebenen Jupyter Notebook. Für die QR-Zerlegung darf `numpy.linalg.qr` benutzt werden. Die Zahl der Iterationen sollte fest auf 100 gesetzt werden.

Aufgabe 3 (Singulärwertzerlegung berechnen, *4 Punkte*)

Im Skript wird im Abschnitt "Berechnung einer Singulärwertzerlegung (SVD)" beschrieben, wie man die Singulärwertzerlegung einer Matrix $A \in \mathbb{C}^{m \times m}$ effizient und robust berechnen kann, indem man die Eigenwertzerlegung der hermiteschen Matrix H berechnet. Wenn $A = U \cdot \Sigma \cdot V^*$ eine Singulärwertzerlegung von A ist, ist

$$H := \begin{pmatrix} 0 & A^* \\ A & 0 \end{pmatrix} = \begin{pmatrix} V & V \\ U & -U \end{pmatrix} \cdot \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \cdot \begin{pmatrix} V & V \\ U & -U \end{pmatrix}^*$$

eine Eigenwertzerlegung von H. Wenn man also die Eigenwertzerlegung von H berechnet und die Eigenwerte und Eigenvektoren absteigend sortiert und normiert, kann man die Singulärwertzerlegung von A direkt ablesen.

Implementiere im gegebenen Jupyter Notebook eine Funktion `ComputeSVD(A)`, die zu einer quadratischen Matrix A die Matrix H konstruiert, ihre Eigenwertzerlegung berechnet und daraus die Singulärwertzerlegung konstruiert. Die Funktion sollte ein Tripel (U, Σ, V) zurückgeben so, dass $A = U \cdot \Sigma \cdot V^*$ die Singulärwertzerlegung von A ist. Die Singulärwerte in Σ sollten dabei, wie üblich, absteigend sortiert sein. Außerdem muss man unbedingt darauf achten, dass die Spalten von U und V normiert sind.

Hinweis: Zur Berechnung der Eigenwertzerlegung kann die Lösung von Aufgabe 2 oder `numpy.linalg.eigh` genutzt werden.

Aufgabe 4 (Program erklären, *0.5+0.5+1+1+1+1=5 Punkte*)

Betrachte das Programm in Listing 1 Bearbeite die folgenden Aufgaben zu dem Programm.

Listing 1: Ein Python-Programm

```
1  from numpy import *
2  from numpy.random import rand
3  from numpy.linalg import norm, inv
4
5  M = rand(4, 4)
6  A = M + M.T
7  #A = diag([3.0, 2.0, -3.0, 0.0])
8
9  u = rand(4, 1)
10 u = u / norm(u)
11
12 m = 1.8
13 B = inv(A - m * eye(4, 4))
14
15 v = u
16 for i in range(100):
17     w = dot(B, v)
18     v = w / norm(w)
19
20
21 l = dot(v.T, dot(A, v))
22
23 print(l)
```

a) Was für Objekte sind `A` und `u` nach Zeile 10? Wie viele Zeilen und Spalten enthalten sie und wie ergeben sich die Einträge?

b) Welche Eigenschaft der Matrix `A` wird durch Zeile 6 garantiert?

c) Gib eine explizite mathematische Formel (in Abhängigkeit von `A`, `m` und `u`) für `w` nach der ersten Schleifeniteration an.

d) Gib eine explizite mathematische Formel (in Abhängigkeit von `A` , `m` und `u`) für `v` in Zeile 20 an.

e) Welcher Algorithmus aus der Vorlesung ist in Zeile 12 bis 21 implementiert? Was ist die Bedeutung von `l` nach Zeile 21?

f) Angenommen man ersetzt Zeile 6 durch den auskommentierten Code in Zeile 7. Was für einen Wert erwarten Sie für die Ausgabe von Zeile 23?

Hinweis zu Python: Hier sind ein paar Ausschnitte aus der Dokumentation von NumPy.

- `numpy.random.rand(d0,d1,...,dn)` Random values in a given shape.
- `numpy.diag(v)` Extract a diagonal or construct a diagonal array.
- `numpy.linalg.inv(a)` Compute the (multiplicative) inverse of a matrix.
- `numpy.linalg.norm(x)` Returns the 2-norm or Euclidean norm of vector `x`.
- `numpy.eye(N,M)` Return a 2-D array with ones on the diagonal and zeros elsewhere.
- `numpy.dot(a,b)` Dot product of two arrays.
  For 2-D arrays it is equivalent to matrix multiplication, and for 1-D arrays to inner product of vectors (without complex conjugation).

Weihnachtsaufgabe 5 (Polarzerlegung, *4 Punkte*)

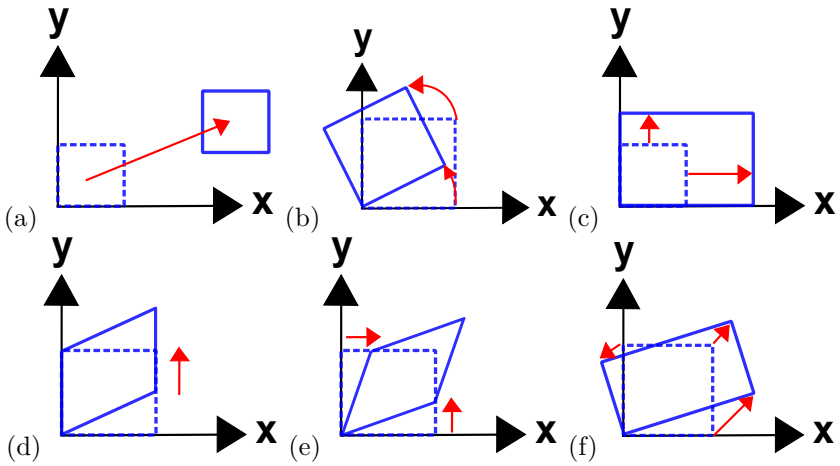

Abbildung 1: Verschiedene Deformationen eines Quadrats

Eine wichtige Frage in der Mechanik ist, ob eine Deformation zu Spannungen führt. Eine reine Translation oder eine reine Rotation führt zu keiner (zusätzlichen) Spannung, während zum Beispiel eine Streckung oder Scherung zu einer Spannung führt, welche berücksichtigt werden muss, wenn es um die Haltbarkeit eines Objekts unter Belastung geht.

Grafik 1 zeigt einige Deformationen. Eine Translation führt offensichtlich zu keiner Spannung, daher betrachten wir im folgenden nur die Deformation des Objekts, welche sich durch den *Deformationsgradienten* $F \in \mathbb{R}^{n\times n}$ beschreiben lässt, welcher die Ableitungen der Komponenten der Einheitsvektoren enthält. Wir haben solche Deformationen bereits kennengelernt als Skalierungs- und Rotationsmatrizen:

$$F_{\text{rot}} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad \text{und} \quad F_{\text{scale}} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$$

Um bei den komplizierteren Deformationen (d), (e), und (f) die Spannung zu bestimmen kann man die Polarzerlegung

$$\begin{aligned} F &= UP \text{ mit} \\ P &:= V\Sigma V^* \\ U &:= WV^* \end{aligned}$$

nutzen wobei $W\Sigma V^*$ die SVD von F ist, U eine unitäre Matrix, und P eine positiv semi-definite hermitesche Matrix ist.

a) Zeige dass P hermitesch und positiv semi-definitit ist und dass U unitär ist.

b) Zeige dass für jede Matrix $F \in \mathbb{C}^{m\times n}$ eine Polarzerlegung existiert.
Hinweis: Verwende für a) und b) die Eigenschaften der SVD

c) Berechne die Polarzerlegung der folgenden Deformationsgradienten (runde auf 3 Nachkommastellen):

$$F_b = \begin{pmatrix} 0.936 & -0.351 \\ 0.351 & 0.936 \end{pmatrix} \qquad F_c = \begin{pmatrix} 2 & 0 \\ 0 & 1.5 \end{pmatrix}$$

$$F_d = \begin{pmatrix} 1 & 0 \\ 0.75 & 1 \end{pmatrix} \qquad F_e = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \qquad F_f = \begin{pmatrix} 1.300 & -0.375 \\ 0.750 & 0.650 \end{pmatrix}$$

(Die SVD muss nicht mehr in Einzelschritten angegeben werden, sondern darf z.B. mit numpy berechnet werden)
Um welchen Winkel wird das Objekt jeweils gedreht?

d) Welche geometrische Bedeutung haben U und P bei der Zerlegung des Deformationsgradienten und was sagt die Determinante über die Deformation aus? Kann man aus der Struktur der Matrix bereits Eigenschaften der Polarzerlegung ablesen?

Weihnachtsaufgabe 6 (Polarzerlegung implementieren, *4 Punkte*)

Neben der Definition über die SVD lässt die Polarzerlegung einer Matrix A sich auch über die folgende Newton-Iteration berechnen:

$$\begin{aligned} U_0 &= A \\ P_0 &= I \\ U_{i+1} &= \frac{1}{2}(U_i + (U_i^*)^{-1}) \\ P_{i+1} &= U_{i+1}^{-1} A \end{aligned}$$

Da P_{i+1} nur von U_{i+1} und A abhängig ist, muss es nicht in jedem Iterationsschritt mit berechnet werden.

a) Implementiere im beigelegten Framework die Polarzerlegung einmal über die SVD von A (hierzu darf `np.linalg.svd` genutzt werden) und einmal über die Newton-Iteration.

b) Plotte die Matrixnorm der Differenz zwischen den zwei Methoden für $\|U_{svd} - U_{it}\|$ und $\|P_{svd} - P_{it}\|$ gegen die Anzahl Iterationen für die Zerlegung einer 100×100 Zufallsmatrix. Mittle um relevante Ergebnisse zu bekommen über die Normen der Zerlegungen von 20 verschiedenen Zufallsmatrizen. Wie viele Iterationen sind nötig, damit der Fehler vernachlässigbar wird?

c) Teste beide Algorithmen für verschiedene Matrixgrößen mit der in Aufgabenteil b) bestimmten Zahl von Iterationen. Plotte wie viel Zeit die Algorithmen jeweils benötigen.
(**Hinweis:** Hierzu kann das Python-Modul `timeit` verwendet werden. Ein Testlauf je Matrixgröße reicht aus)

d) Die exakte Matrix U der Polarzerlegung ist unitär. Wie verhält sich der Fehler, wenn in der Iteration $P_{i+1} = U_{i+1}^* A$ verwendet wird?

Welchen Vorteil hat die Berechnung über die Newton-Iteration?

Frohe Weihnachten!