MATHEMATISCHES INSTITUT
PROF. DR. ACHIM SCHÄDLE
MARINA FISCHER

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

05. FEBRUAR 2019

Schriftliche Prüfung zur Computergestützten Mathematik zur linearen Algebra

Erste Klausur

Aufgabe 1: (2 + 4 Punkte)

Für einen Parameter $k \in \mathbb{N}$ ist die reellwertige Funktion f_k definiert durch

$$f_k(x) = \frac{1}{2}\left[\left(x + i\sqrt{1-x^2}\right)^k + \left(x - i\sqrt{1-x^2}\right)^k\right].$$

Hierbei ist $i = (-1)^{\frac{1}{2}}$ die imaginäre Einheit. (In PYTHON ist die imagnäre Einheit `1j` bzw. `(-1)**(1/2)`.)

(a) Schreiben Sie in die Datei `aufgabe1.py` eine PYTHON-Funktion `myfun(x,k)`, die zu einem Parameter `k` $\in \mathbb{N}$ und einem Array `x` $\in \mathbb{R}^n$ die Funktion auswertet und einen Array `y` $= f_{\mathtt{k}}(\mathtt{x}) \in \mathbb{C}^n$ zurückgibt.

(b) Ergänzen Sie die Datei `aufgabe1.py`, sodass für `k` $= 1, 2, 3, 4, 5$ der Realteil der Funktionen $f_{\mathtt{k}}$ in einem gemeinsamen Fenster auf dem Intervall $[-1, 1]$ gezeichnet werden.

Geben Sie für jedes `k` eine andere Farbe vor und ergänzen Sie das Skript so, dass eine Legende, die die einzelnen Funktionen beschreibt, in Ihrer Graphik angezeigt wird. Platzieren Sie die Legende in der rechten oberen Ecke des Fensters.

Aufgabe 2: (3 + 3 Punkte)

Gegeben sind zwei Polynome a_ℓ und b_ℓ in x, die rekursiv durch

$$\begin{aligned} a_{-1}(x) &= 1, & a_0(x) &= 2 + x, & a_{\ell+1}(x) &= 2(2+x)a_\ell(x) - x^2 a_{\ell-1}(x), \\ b_{-1}(x) &= 0, & b_0(x) &= 2, & b_{\ell+1}(x) &= 2(2+x)b_\ell(x) - x^2 b_{\ell-1}(x) \end{aligned}$$

für $\ell = 0, 1, \ldots$ definiert sind.

(a) Schreiben Sie eine Funktion `rationalfun(x,k)` in die Datei `aufgabe2.py`, die zu gegebenem Array `x` $\in \mathbb{R}^n$ und einer natürlichen Zahl `k` die Werte der rationalen Funktion $p_k(x) := a_k(x)/b_k(x)$ in einem Array der Länge n ausgibt.

(b) Ergänzen Sie das PYTHON-Skript `aufgabe2.py`, sodass die Funktion p_k in einem Fenster auf dem Intervall $[0.1, 20]$ für `k` $= 0$, `k` $= 2$ und `k` $= 6$ graphisch darstellt wird. Verwenden Sie für `x` hierbei $n = 30$ äquidistante Werte zwischen 0.1 und 20 und beschränken Sie den y-Achsenabschnitt auf das Intervall $[1, 10]$.
Zeichnen Sie $p_{\mathtt{k}}$ für `k` $= 2$ mit einer roten durchgezogenen Linie und für `k` $= 6$ mit einer grünen gestrichelten Linie.

Aufgabe 3: (6 Punkte)

Für eine symmetrische, positiv definite $(n \times n)$-Matrix $A = (a_{ij})_{i,j=0}^{n-1}$ kann man eine Zerlegung $A = LDL^T$ mit einer linken unteren Dreiecksmatrix L, die Einsen aus der Diagonalen hat, und einer Diagonalmatrix D berechnen. Ein Verfahren dazu ist durch den folgenden Pseudocode gegeben.

Eingabe : A
Ausgabe: L, D, `flag`
for k *von* 0 *bis* $n-1$ **:**
  if $a_{kk} = 0$ **:**
    setze `flag`$=k$ und brich ab
  $d_k = a_{kk} - \sum_{i=0}^{k-1} d_i l_{k,i}^2$
  for j *von* $k+1$ *bis* $n-1$ **:**
    $a_{jk} = a_{jk} - \sum_{i=0}^{k-1} d_i a_{ki} a_{ji}$
    $a_{jk} = a_{jk}/d_k$
L besteht aus dem linken unteren Teil des überschriebenen A unterhalb der Diagonalen und den Einträgen 1 auf der Hauptdiagonalen.
D ist eine Diagonalmatrix mit den Einträgen d_j.
`flag` gibt an, bei welcher Zeile/Spalte eine 0 auf der Diagonalen von A stand und ob der Algorithmus abgebrochen wurde.

In der Datei `aufgabe3.py`, die oben angegebene Pseudcode umsetzt, haben sich vier Fehler eingeschlichen. Finden, korrigieren und kommentieren Sie diese Fehler sinnvoll.

Ergänzen Sie die Datei `aufgabe3.py` um drei geeignet gewählte Beispiele, um Ihre korrigierte LDL Zerlegung zu testen. Hierbei soll mindestens einmal ein Ergebnis mit `flag=None` und ein weiteres, in dem `flag` einen anderen Wert annimmt, vorkommen.

Handschriftliche Korrekturen in der unten abgedruckten Datei werden bei der Bewertung NICHT berücksichtigt. Eine schreibgeschützte Kopie des unten angegebenen Codes finden Sie in der Datei `LDL.py`.

```
1  def ldl(A):
2      A = A.astype('float').copy()
3      m, n = A.shape
4      flag = None
5      assert (m == n), "Matrix muss quadratisch sein"
6      assert (np.all(A = A.T)), "Matrix nicht symmetrisch"
7      D = np.zeros(n)
8      for k in range(n):
9          if A[k, k] == 0:
10             flag = k
11             break
12         D[k] = A[k, k]
13         for i in range(k):
14             D[k] = D[k] -  D[i]*A[k, i]*2
15         for j in range(k, n):
16             A[j, k] = A[j, k]
17             for i in range(k):
18                 A[j, k] = A[j, k] - D[i]*A[k, i]*A[j, i]
19             A[j, k] = A[j, k]*D[k]
20     return np.tril(A, k=-1) + np.eye(n), np.diag(D), flag
```

Aufgabe 4: (2 + 1 + 3 Punkte)

Für eine natürliche Zahl n sei $\mathbb{P}_n = \left\{\sum_{j=0}^{n} c_j x^j \ : \ c_j \in \mathbb{R}\right\}$ der $n+1$ dimensionale Vektorraum der Polynome vom Grad kleiner gleich n ausgestattet mit dem Skalarprodukt

$$\langle p, q\rangle = \int_{-1}^{1} p(t)\, q(t)\, \mathrm{d}t.$$

Dieses Skalarprodukt ist als `SProdL2` in der Polynomklasse `polynompy.py` implementiert. Die Legendrepolynome $(L_0, L_1, L_2, \ldots, L_n)$, die ebenfalls in der Polynomklasse implementiert sind, sind eine Orthogonalbasis dieses Vektorraums.

(a) Um ein gegebenes Polynom p in dieser Basis darzustellen, d.h. Koeffizienten $a_0, \ldots, a_n$ zu finden, sodass $p = \sum_{j=0}^{n} a_j L_j$ gilt, berechnet man diese Koeffizienten durch

$$a_j = \frac{\langle p, L_j\rangle}{\langle L_j, L_j\rangle},\ j = 0, \ldots, n.$$

Schreiben Sie in die Datei `aufgabe4.py` eine Funktion `LegendreKoeff(p, n)`, die die obigen Koeffizienten für ein gegebenes Polynom p berechnet und in einem Array der Länge $n+1$ ausgibt.

Die Abbildung $\alpha : p \mapsto \frac{d}{dx}p$ ist eine lineare Abbildung von $\mathbb{P}_n$ in $\mathbb{P}_n$.

(b) Plotten Sie das zweite Legendrepolynom L_2 und $\alpha(L_2)$ mit Hilfe der `plot`-Methode der Polynomklasse im Intervall $[-1, 1]$ in ein gemeinsames Bild.

(c) Schreiben Sie eine Funktion `alphaMatrix(n)`, die für eine natürliche Zahl n die Darstellung der Abbildung α als $(n+1) \times (n+1)$-Matrix (Array) bezüglich der Legendrebasis in Urbild- und Bildraum berechnet.

Aufgabe 5: (1 + 1 + 2 + 2 Punkte)

Implementieren Sie in der Datei `aufgabe5.py` eine Funktion `plotEv(A)`, die für eine gegebene symmetrische (2×2)-Matrix A

(a) die zugehörigen Eigenwerte und Eigenvektoren mit Hilfe von `numpy.linalg.eig` berechnet,

(b) die Eigenvektoren auf die euklidische Länge 1 normiert,

(c) die Menge $\left\{Au \ : \ ||u||_2 = 1,\ u = (x, y)^T \in \mathbb{R}^2\right\}$ (Bild des Einheitskreises) graphisch darstellt und

(d) die mit den zugehörigen Eigenwerten skalierten normierten Eigenvektoren in das Bild aus (c) einzeichnet. Hierfür reicht eine Linie, die den Ursprung mit dem Punkt $(\lambda_j x_j,\ \lambda_j y_j)$ verbindet, wobei λ_j der Eigenwert zum Eigenvektor $(x_j, y_j)^T$ ist.

Aufgabe 6: (4 + 2 Punkte)

In einem Experiment wurden die folgenden Daten gemessen und in der Datei `aufgabe6daten.dat` abgespeichert:

x_i	0.34	0.61	0.92	1.05	1.80	2.62	2.76	2.99	3.65
y_i	1.35	0.65	0.29	0.19	1.86	0.29	0.09	-0.34	0.73

Es wird angenommen, dass die Daten für drei Parameter α_1, α_2 und α_3 folgende Gesetzmässigkeit erfüllen

$$f(x) = \alpha_1 \cos(\pi x) + \alpha_2 \sqrt{x} + \alpha_3 x.$$

(a) In dem PYTHON-Skript `aufgabe6.py`, sollen die Messdaten aus `aufgabe6daten.dat` eingelesen und die Parameter α_1, α_2 und α_3 bestimmt werden, sodass die Funktion f in den Punkten x_i die Werte y_i im Sinne der kleinsten Fehlerquadrate bestmöglich approximiert. Lösen Sie das kleinste Fehlerquadrate Problem mit Hilfe der QR-Zerlegung `numpy.linalg.qr` und dem Befehl `numpy.linalg.solve`.

Berechnen Sie darüber hinaus den Fehler

$$err = \left(\sum_{i=1}^{9} |y_i - f(x_i)|^2 \right)^{1/2}$$

und geben Sie ihn aus.

(b) Ergänzen Sie das Skript `aufgabe6.py`, sodass Ihre Ergebnisse aus Aufgabenteil (a) graphisch darstellt werden. Ihre Graphik sollte

- die Datenpunkte (x_i, y_i) als schwarze $*$ und
- die Funktion f, mit den in (a) berechneten Parametern α_1, α_2 und α_3, als rote durchgezogene Linie auf dem Intervall $[0, 4]$ enthalten.

Ergänzen Sie Ihre Graphik um eine aussagekräftige Legende.