Probeklausur

Dies ist eine Probeklausur.

(a) Lösen Sie jede Aufgabe in der dafür vorgesehenen Python-Datei (Aufgabe 1 in A1.py, Aufgabe 2 in A2.py usw.). Sollte es Teilaufgaben geben, so schreiben Sie Ihre Lösung bitte in den dafür vorgesehenen Abschnitt.

(b) Sollten Sie eine Teilaufgabe nicht schaffen die Sie für einen anderen Teil benötigen, so können Sie (oft) die in der Datei befindliche Backuplösung verwenden.

(c) Verändern Sie NICHT die Kommentare die schon in den Dateien stehen und fügen Sie keine zusätzlichen #%%-Kommentare hinzu.

(d) Ihre Lösung sollte mittels des „run“-Befehls (F5) lauffähig sein.

(e) Sie dürfen alle Dateien in diesem Ordner (Vorlesung und Python-Befehl-Liste) und ein doppelseitig handbeschriebenes DIN A4 Blatt zum Lösen verwenden. In diesem Ordner befindet sich die html-Seite *Vorlesung.html*, in der alle Vorlesungen aufgelistet sind.

Aufgabe 1 (*Funktionen und Plots*)

(a) Schreiben Sie eine PYTHON-Funktion `y = f(x)`, die

$$f(x) = \frac{1}{4(x-2)^2}$$

auf dem `array` `x` auswertet.

(b) Schreiben Sie eine PYTHON-Funktion `y = g(x)`, die die stückweise definierte Funktion

$$g(x) = \begin{cases} (x-1)^2, & x \leq 1 \\ 1 + \sin(\frac{\pi}{2} + \pi x), & x > 1 \end{cases}$$

auf dem `array` `x` auswertet.

(c) Plotten Sie beide Funktionen in einen Plot im Intervall $[0, 4]$. Achten Sie darauf, dass $g(x)$ auch an der wichtigen Stelle $x = 1$ ausgewertet wird. Schränken Sie die y-Achse auf das Intervall $[0, 2]$ ein.
Die Funktion $f(x)$ soll in einer grünen, gestrichelten Linie und $g(x)$ in einer schwarzen, durchgezogenen Linie geplottet werden.
Erstellen Sie Ihre Legende so, dass sie sich in der oberen linken Ecke befindet.

Aufgabe 2 (*Legendre-Polynome*)

Die *integrierten Legendre-Polynome* $L_k(x)$ sind wie folgt definiert:

$$(k+1)L_{k+1}(x) = (2k-1)xL_k(x) - (k-2)L_{k-1}(x) \text{ für } k \geq 2$$

mit

$$L_1(x) = x \text{ und } L_2(x) = \frac{1}{2}(x^2 - 1).$$

(a) Schreiben Sie eine Funktion `y = lepore(n, x)`, die das `n`-te integrierte Legendre-Polynom an den Stellen des `array`s `x` auf rekursive Weise auswertet. Printen Sie eine aussagekräftige Warnung aus, falls `n` keine ganze Zahl ist und geben Sie dann `None` zurück.

(b) Plotten Sie für $n = 2, 3, \ldots, 6$ die Polynome $L_n(x)$ mit Hilfe einer Schleife gut sichtbar im Intervall $[-1, 1]$.

Hinweis: Sollten Sie (a) nicht gemacht haben, können Sie in (b) stattdessen die in der PYTHON-Datei definierte Funktion `lepodi(n, x)` verwenden.

Aufgabe 3 (*Elimination / Fehler im Code*)

Gegeben sei folgender Quelltext für die Eliminierung der Einträge der ersten Spalte einer Matrix `A` $\in \mathbb{R}^{\mathtt{m}\times\mathtt{n}}$. Eliminiert wird mit einem betragsgrößten Element der Spalte, das zuvor nach oben getauscht wird. `A` sei vom `type array` mit `float`-Werten. Zurückgegeben werden sollen die Matrix, die Faktoren für die Elimination und der ursprüngliche Index der nach oben getauschten Zeile.

```
1 def EliminiereSpalte(A):
2     m, n = A.shape;
3     index = np.argmax(abs[A[:, 0]]);
4     A[[0, index], :] = A[[0, index], :];
5     l = A[1:, [ 0 ] ] * A[0, 0];
6     A[1:, :] = A[1:, :] - l @ A[[ 0 ], :];
7
8     return A, l, index;
```

(a) Finden und korrigieren Sie die 3 Fehler im Quelltext. Markieren Sie die gefundenen Stellen mit Kommentaren.

(b) Kommentieren Sie, zusätzlich zu den Kommentaren aus (a), die Funktion Zeile für Zeile.

Aufgabe 4 (*Pseudocode*)

Befehle: `la.solve`, `la.eig`
Sei $m \geq 1$ und $A \in \mathbb{R}^{m\times m}$ symmetrisch und invertierbar. Gesucht ist der betragsmäßig kleinste Eigenwert von A und ein zugehöriger Eigenvektor. Dazu dient der folgende Algorithmus:

`Gegeben sei eine Toleranz` $\varepsilon > 0$. `Wähle den Startwert` $v_0 = [1, \ldots, 1]^T \in \mathbb{R}^m$.
`Für` $n = 1, 2, \ldots$
    `Setze` $\tilde{v}_n = A^{-1} v_{n-1}$ `und` $v_n = \tilde{v}_n / \|\tilde{v}_n\|_2$
    `Setze` $\mu_n = (v_n)^T A v_n$
    `Brich ab, falls` $\|Av_n - \mu_n v_n\|_2 < \varepsilon$

(a) Schreiben Sie eine Funktion `mu, v = invPot(A, eps)`, die den obigen Algorithmus implementiert. `A` ist die Matrix A als `array`, `eps` die Toleranz für das Abbruchkriterium. `mu` und `v` sind der approximierte Eigenwert bzw. ein dazugehöriger normierter Eigenvektor. Geben Sie die Anzahl der benötigten Iterationen mit `print` aus.

(b) Setzen Sie `eps` $= 10^{-5}$ und testen Sie Ihre Implementierung an der Matrix

$$A = \mathtt{np.diag}([\mathtt{10}]*\mathtt{19} + [\mathtt{1}]) \in \mathbb{R}^{20\times 20}.$$

Vergleichen Sie das Ergebnis mit dem von `la.eig`.

Aufgabe 5 (*Kontrast verändern*)

(a) Schreiben Sie eine Funktion `bildk = kontrast(bild, k)`, welche ein Graustufenbild `bild` als `array` als Parameter erhält und mittels der Abbildung

$$f_k(x) = (\tfrac{1}{2} + \tfrac{1}{2}\cos(\pi + \pi x))^k$$

den Kontrast im Bild `bild` verändert, indem die Helligkeit der Pixel $x \in [0, 1]$ verändert wird. Das Ergebnis soll in `bildk` zurückgegeben werden.

(b) Importieren Sie das Graustufenbild `Foto.png` aus Ihrem Verzeichnis und speichern Sie es in der Variablen `daten`.

(c) Testen Sie Ihre Funktion `kontrast(bild, k)` mit dem Bild `daten`, indem Sie das Originalbild und das Kontrastveränderte in Grau untereinander plotten. Verwenden Sie hierzu `k` = 1,5.

Hinweis: Sollten Sie (a) und/oder (b) nicht gemacht haben, können Sie stattdessen auch die im Notebook definierten `verschieben_sicher(bild, k)` und/oder `bild_sicher` verwenden.

Aufgabe 6 (*Lineare Ausgleichsrechnung*)

Befehle: `la.qr(..., mode = 'economic')`, `la.solve_triangular`

Gegeben sind die x-Werte `x` und die gestörten Funktionswerte `y`, die Sie in Ihrem JUPYTER-Notebook zu dieser Aufgabe finden.
Aus sicherer Quelle wissen Sie, dass die Werte gestörte Auswertungen der Funktion

$$y = f(x) = \lambda_1 \log(x) + \lambda_2 \sin(2 \cdot (1 - x)) + \lambda_3 x$$

sind, wobei Sie die Koeffizienten λ_1, λ_2 und λ_3 nicht kennen. Dabei sei log = ln der natürliche Logarithmus.

Gesucht sind Koeffizienten $\widetilde{\lambda}_1$, $\widetilde{\lambda}_2$ und $\widetilde{\lambda}_3$, welche die ungestörten Koeffizienten λ_1, λ_2 und λ_3 von f im Sinne der kleinsten Fehlerquadrate möglichst gut approximieren.

(a) Stellen Sie die Matrix A und den Vektor b des zugehörigen Minimierungsproblems der Form

$$\min_{\widetilde{\lambda} \in \mathbb{R}^3} \|A\widetilde{\lambda} - b\|_2 \tag{1}$$

auf. Dabei sei $\widetilde{\lambda} = [\widetilde{\lambda}_1, \widetilde{\lambda}_2, \widetilde{\lambda}_3]^T$.

(b) Bestimmen Sie mit Hilfe der reduzierten QR-Zerlegung von A einen Vektor $\widetilde{\lambda}$, der (1) löst. Lösen Sie nicht die Normalengleichung.

(c) Plotten Sie die Funktion $\widetilde{f}(x)$, welche die Koeffizienten $\widetilde{\lambda}_i$ statt λ_i in f verwendet, im Intervall $[1, 20]$ zusammen mit den gestörten Funktionswerten in einen gemeinsamen Plot. $\widetilde{f}$ soll an mindestens 100 Punkten im Intervall ausgewertet werden.

Der Graph der Approximation $\widetilde{f}$ soll in Blau und die gestörten Funktionswerte sollen als rote Punkte („`point markers`“) dargestellt werden.

Hinweis: Sollten Sie (a) nicht gemacht haben, können Sie die in der PYTHON-Datei definierte Matrix `B` verwenden.