Heinrich Heine Universität Düsseldorf

Mathematisches Institut
Prof. Dr. Achim Schädle
Melinda Hagedorn
Dr. Marina Fischer
Dr. Christoph Matern
24. Mai 2023

| 22 | 23 | 24 | Σ |
|---|---|---|---|
| | | | |

Name: ____________________
Name: ____________________

# Numerik I – 8. Übungsblatt

**Hinweis:** Verwenden Sie als Dateiname für die schriftliche Abgabe bitte
`Name1_Name2_Blatt8.pdf` bzw. `Name1_Blatt8.pdf`,
wobei Name1 (und Name2) Ihre Nachnamen sind. Dies erleichtert die Zuordnung Ihrer Abgaben.

## Aufgabe 22:

(a) Für $f(x) = x^k$, $k > 1$, bestimmen Sie das Hermite-Interpolationspolynom $p \in \mathcal{P}_3$ mit

$$p(0) = f(0) \qquad p(1) = f(1)$$
$$p'(0) = f'(0) \qquad p'(1) = f'(1).$$

(b) Berechnen Sie $p'(1/2)$ mit Hilfe der numerischen Differentiation wie in der Vorlesung.

(c) Für welche Werte von $k$ würden Sie der Approximation $p'(1/2) \approx f'(1/2)$ vertrauen?

## Aufgabe 23:

(a) Bestimmen Sie mit Hilfe des Newtonschen dividierten Differenzenschemas das Polynom kleinsten Grades $q$, das die Daten $x_1 = -1$, $f(x_1) = 2$ und $x_2 = 1$, $f(x_2) = 4$ interpoliert (d.h. $q(x_1) = 2$ und $q(x_2) = 4$) und zusätzlich $q''(x_2) = 2$ erfüllt.

(b) Geben Sie $q'(x_2)$ an, *ohne* $q(x)$ abzuleiten.

## Aufgabe 24:

Berechnen Sie per Hand die LR Zerlegung der Matrix

$$A = \begin{pmatrix} 2 & 0 & 1 \\ -4 & -3 & -3 \\ 6 & 6 & 3 \end{pmatrix}$$

ohne Pivotisierung, d.h. ohne Vertauschung von Zeilen.

**b.w.**

**Programmieraufgabe 8:** (Lineares Gleichungssystem für Computertomographie)

Dateiname: `Name1_Name2_A8.py` bzw. `Name1_A8.py`,
wobei Name1 (und Name2) Ihre Nachnamen sind. Dies erleichtert die Zuordnung Ihrer Abgaben.
Wir betrachten ein reguläres Gitter über $\Omega = [-1, 1]^2$, d.h. eine Zerlegung in $K$ Quadrate mit Kantenlänge $2/\sqrt{K}$. Weiterhin seien für Winkel $\varphi_j$ und Verschiebungen $s_j$, $j = 0, \dots, J-1$ die Strahlen

$$L_j(t) = s_j\omega(\varphi_j) + t\omega^\perp(\varphi_j), \qquad \omega(\varphi) = (\cos(\varphi), \sin(\varphi)), \quad \omega^\perp(\varphi) = (-\sin(\varphi), \cos(\varphi))$$

gegeben.

Der $(j,k)$-Eintrag der Matrix des linearen Gleichungssysten aus Beispiel (15.1) der Vorlesung entspricht der Länge des Schnittes der $k$-ten Gitterzelle mit dem $j$-ten Strahl.

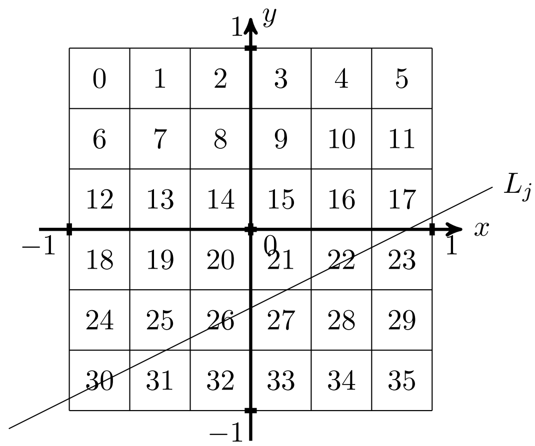


Abbildung 1: Beispiel für ein reguläres Gitter von $\Omega$ ($K = 6^2 = 36$) und für einen Strahl $L_j$ mit den Zellen $k = 0, \dots, 35$.

(a) Schreiben Sie eine Methode `getTomographyMatrix(K, phi, s)`, die die Matrix für das lineare Gleichungssystem aus Beispiel (15.1) der Vorlesung aufstellt und zurückgibt.
Dabei ist `K` die Anzahl der Quadrate des Gitters (als `int`) und `phi` und `s` sind `arrays` der Länge $J$, deren $j$-te Einträge den $j$-ten Strahl beschreiben.

(b) Testen Sie Ihre Methode folgendermaßen:
Die Datei `Tomographiedaten.zip` (Vorlesungseite) enthält die Dateien `daten?x?.npz` mit Parametern `K`, `phi` und `s` für die Matrix und Messdaten `d` (als `array`).

Verwenden Sie für jeden Datensatz jeweils die zugehörige Matrix aus Aufgabenteil (a) um das Gleichungssystem für das Tomographieproblem aufzustellen und mit den Daten `d` und `scipy.linalg.lstsq` zu lösen. Zeichnen Sie die generierten Bilder in einzelne Plots.

*Hinweise*:

- Zur zip-Datei: Importieren Sie das Modul `tomograpiemodul.py`. Es beinhaltet die Methode `lade_daten`, die Ihnen die Daten aus `daten?x?.npz` gibt und die Methode `zeichne_bild`, die Ihnen das Bild schön zeichnet.
Zum Vergleich: `bild2x2.png` ist die Lösung der Daten aus `daten2x2.npz`.

- Die ersten Bilder (bis `daten20x20.npz`) sollte Ihr Computer ohne Probleme berechnen können. Diese genügen für die Punkte dieser Aufgabe. Etwa ab `daten60x60.npz` werden Sie ohne `sparse`-Formate aus `scipy.sparse` (d.h. dass keine unnötigen Nullen gespeichert werden) Schwierigkeiten bekommen.

**b.w.**

**Programmieraufgabe 9:**

Dateiname: `Name1_Name2_A9.py` bzw. `Name1_A9.py`,
wobei Name1 (und Name2) Ihre Nachnamen sind. Dies erleichtert die Zuordnung Ihrer Abgaben.

(a) Schreiben Sie eine Methode `spline_not_a_knot(x, f)` zur Berechnung der Parameter $\tau_i$, $0 \le i \le n$ eines kubischen Splines $s$ mit $s(x_i) = f(x_i)$, $0 \le i \le n$ und den not-a-knot Randbedingungen $s|_{[x_0,x_2]} \in C^3$ und $s|_{[x_{n-2},x_n]} \in C^3$.
`x` und `f` sind jeweils `array`s mit den $x_i$ bzw. $f(x_i)$ als Einträgen. Die $\tau_i$ sollen in einem `array` zurückgegeben werden.

(b) Schreiben Sie weiterhin eine Methode `spline_eval(x, f, Tau, X)` welches einen kubischen Spline $s$ mit den Parametern $\tau_i$, $0 \le i \le n$ auf vorgegebenen Punkten $X_j$, $0 \le j \le N \in \mathbb{N}$ auswertet.
`x` und `f` sind gegeben wie in (a), `Tau` ist das `array` mit den $\tau_i$ (diese (a)) und `X` ist ein `array` mit den auszuwertenden Stellen $X_j$. Die Auswertungen $s(X_j)$ sollen in einem `array` zurückgegeben werden.

(c) Testen Sie Ihre Methoden an der Funktion

$$f(x) = e^{-x^2/2}$$

Stellen Sie dazu jeweils die Funktion $f$ zusammen mit den not-a-knot-Spline-Interpolierenden in den Knoten $x_i = -5 + 10i/n$, $0 \le i \le n$ für $n = 5, 10$ un $20$ graphisch dar. Verwenden Sie zur Darstellung $N \ge 200$ Punkte.

**Hinweis:** Das lineare Gleichungssystem können Sie mit `scipy.linalg.solve` lösen.
Die Methode `scipy.interpolate.CubicSpline` kann den not-a-knot-Spline auswerten. Sie können diese verwenden, um Ihr Ergebnis zu überprüfen. Ein Wrapper zu dieser (oder einer ähnlichen) Methode ist für die Lösung der Aufgabe nicht ausreichend!

**Bemerkungen:** Das sind die beiden nächsten Programmieraufgaben. Auf Blatt 9 gibt es dann keine Programmieraufgabe(n).

**Abgabe der Übungsaufgaben am Mittwoch, 31. Mai bis 10:30 Uhr per ILIAS unter „Übungen zu Numerik I“.**
**Abgabe der Programmieraufgabe 9 am Mittwoch, 31. Mai bis 10:30 Uhr per ILIAS unter „Programmierübungen zu Numerik I“.**
**Abgabe der Programmieraufgabe 8 am Mittwoch, 7. Juni bis 10:30 Uhr per ILIAS unter „Programmierübungen zu Numerik I“.**