Wissenschaftliches Rechnen

Aufgabenblatt 1 (Praxis)

wr@isg.cs.uni-magdeburg.de

SoSe 2023

Allgemeine Hinweise:

- Die Aufgaben sind von jeder/m Studierenden *einzeln* zu bearbeiten und abzugeben (Plagiate werden entsprechend der Studienordnung geahndet).
- Verwenden Sie den vorgegebenen Skelett-Code. Die zu implementierenden Funktionen befinden sich in der Datei **main.py**. Ihr Code ist an den mit `# TODO: ...` gekennzeichneten Stellen einzufügen. Die `NumPy`-Funktionen, welche Sie zur Lösung einer Aufgabe nicht verwenden dürfen, sind unter `Forbidden` in der Docstring Beschreibung der entsprechenden Funktion aufgelistet.
- Bitte reichen Sie `main.py` mit Ihrem Lösungen bis **Freitag, den 09.06.2023, um 18:00 Uhr** über Ihr git repository ein.

"Spalten"-Matrizen und Vektoren in `NumPy`

In der Mathematik sind 1-dimensionale "Spalten"-Matrizen $A \in \mathbb{R}^{n\times 1}$ identisch zu Vektoren $v \in \mathbb{R}^n$. In `NumPy` werden diese jedoch als unterschiedliche Objekte repräsentiert. Eine $(5,1)$-Matrix $A \in \mathbb{R}^{5\times 1}$ mit Nulleinträgen wird zum Beispiel durch `A = np.zeros((5,1))` erzeugt, wohingegen ein 5-Vektor $v \in \mathbb{R}^n$ durch `v = np.zeros(5)` erzeugt wird. Dieser Unterschied ist ersichtlich, wenn man die Größe der Objekte betrachtet:

```
>>> A.shape
(5, 1)
>>> v.shape
(5,)

```

Bei der Durchführung von Berechnungen in Numpy sollte der Unterschied beachtet werden.

Aufgabe 1: Gaußsches Eliminationsverfahren mit Pivoting (3 Punkte)

In dieser Aufgabe soll die Lösung von linearen Gleichungssystemen durch das Gaußsche Eliminationsverfahren und Rückwärtseinsetzen implementiert werden.

Aufgabe 1.1: Gauß-Eliminierung (2 Punkte)

Implementieren Sie das Gaußsche Eliminationsverfahren sowohl ohne als auch mit Pivoting in der Funktion `gaussian_elimination()`. Wenn die Eingabedaten inkompatible Größen haben, die Matrix nicht quadratisch oder das Gleichungssystem ohne Pivoting nicht lösbar ist, dann soll ein `ValueError` erzeugt werden. In `tests.py` gibt es bereits einen Test, welcher Ihre Implementierung mit zufällig erzeugten Eingabedaten testet.

Aufgabe 1.2: Rückwärtseinsetzen (1 Punkt)

Implementieren Sie Rückwärtseinsetzen in der Funktion `back_substitution()`. Die Funktion bekommt eine obere Dreiecksmatrix und einen Vektor als Argumente übergeben. Wenn die Argumente der Funktion inkompatibel sind (die Größen nicht übereinstimmen) oder wenn es keine/unendlich viele Lösungen beim Einsetzen gibt, soll ein `ValueError` erzeugt werden.

Aufgabe 2: Lösung von Gleichungssystemen mit der Cholesky-Zerlegung (3 Punkte)

Aufgabe 2.1: Berechnung der Cholesky-Zerlegung (2 Punkte)

Implementieren Sie die Cholesky-Zerlegung $A = LL^T$ einer Matrix $A \in \mathbb{R}^{n \times n}$ in der Funktion `compute_cholesky()`. Ihre Funktion soll die untere Dreiecksmatrix L mit der in der Vorlesung vorgestellten Formel berechnen und zurückgeben. Für Eingabematrizen, die nicht die Voraussetzungen für die Cholesky-Zerlegung erfüllen, soll ein `ValueError` erzeugt werden.

Aufgabe 2.2: Lösung von Gleichungssystemen (1 Punkt)

Mithilfe der Cholesky-Zerlegung $A = LL^T$ können lineare Gleichungssysteme $Ax = b$ effizient gelöst werden. Implementieren sie die Funktion `solve_cholesky()`, welche einen Cholesky Faktor L und den Vektor b als Parameter übergeben bekommt und die Lösung x des Gleichungssystems $LL^Tx = b$ zurückgibt. Für die Implementierung dürfen sie die Funktion `back_substitution` aus Aufgabe 1.2 des Aufgabenblatts verwenden. Überprüfen sie, ob die Eingabematrix L tatsächlich eine quadratische, untere Dreiecksmatrix ist und erzeugen sie einen `ValueError`, falls dies nicht der Fall ist.

Aufgabe 3: Lineare Ausgleichsrechnung (4 Punkte)

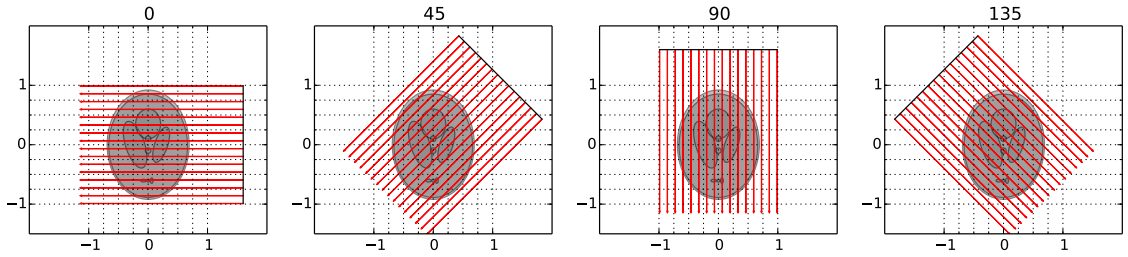


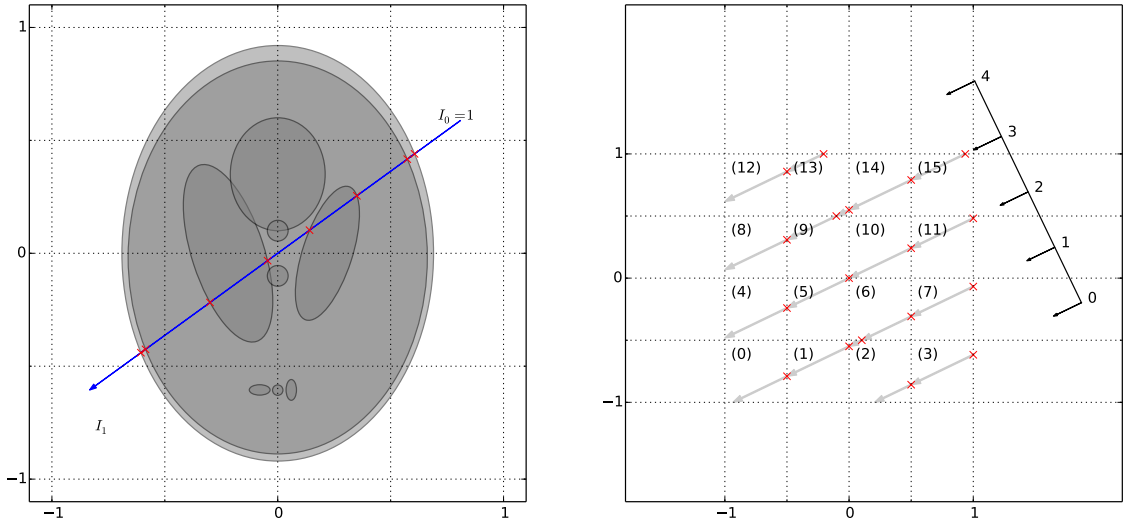
Abbildung 1: Bei der Computertomographie werden aus verschiedenen Winkeln eine Reihe paralleler Röntgenstrahlen ausgesendet und der Intensitätsverlust der Strahlen gemessen, welcher beim Durchdringen des Objektes auftritt.

In dieser Aufgabe implementieren Sie den Rekonstruktionsschritt eines Tomographen, welcher die Dichten ρ_{ij} in den Zellen eines 2D-Volumens bestimmt. Diese Dichten können als ein sogenanntes tomographisches Bild dargestellt werden, siehe Abb. **??**. Um die Eingangsdaten zu bestimmen, steht Ihnen die Funktion `take_measurements()` zur Verfügung, welche jeweils eine Messung des Tomographen aus einem Winkel θ durchführt (siehe Abb. **??**) und ihnen sowohl die (log-)Intensitäten V_m der m parallelen Strahlen nach dem Durchgang durch das Volumen zurückgibt als auch die Länge L_{ij}^m der Strecke, welche jeder Strahl durch die (i, j)-te Zelle zurücklegt, siehe Abb **??**. Um eine robuste Lösung zu erhalten, müssen Messungen aus verschiedenen Richtungen durchgeführt werden und es müssen mehr Messwerte als Unbekannte daraus resultieren. Kombiniert man die Daten aus den Messungen, so kann das tomographische Rekonstruktionsproblem durch folgendes lineares Gleichungssystem beschrieben werden:

$$\underbrace{\begin{pmatrix} L_{11}^{(1)} & L_{12}^{(1)} & \cdots & L_{1n}^{(1)} & L_{21}^{(1)} & L_{22}^{(1)} & \cdots & L_{nn}^{(1)} \\ L_{11}^{(2)} & L_{12}^{(2)} & \cdots & L_{1n}^{(2)} & L_{21}^{(2)} & L_{22}^{(2)} & \cdots & L_{nn}^{(2)} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ L_{11}^{(K)} & L_{12}^{(K)} & \cdots & L_{1n}^{(K)} & L_{21}^{(K)} & L_{22}^{(K)} & \cdots & L_{nn}^{(K)} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} \varrho_{11} \\ \varrho_{12} \\ \vdots \\ \varrho_{1n} \\ \varrho_{21} \\ \varrho_{22} \\ \vdots \\ \varrho_{nn} \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_K \end{pmatrix}}_{b} . \tag{1}$$

Aufgabe 3.1: Aufstellen des linearen Systems (2 Punkte)

Stellen Sie mit Hilfe der Funktion `setup_system_tomograph()` das überbestimmte lineare Gleichungssystem in Gl. **??** auf. Benutzen Sie dazu `take_measurement()`, um `n_shot` Messungen aus äquidistant in $[0, \pi)$ verteilten Richtungen durchzuführen (`n_shot` sollte keine Potenz von zwei sein). Achten Sie



(a) (b)

Abbildung 2: Links: Abschwächung der Intensität eines Strahles beim Durchgang durch ein Volumen. Dies wird verwendet, um die Dichte über einem diskreten Gitter zu rekonstruieren. Rechts: Länge von verschiedenen, parallelen Strahlen in Gitterzellen.

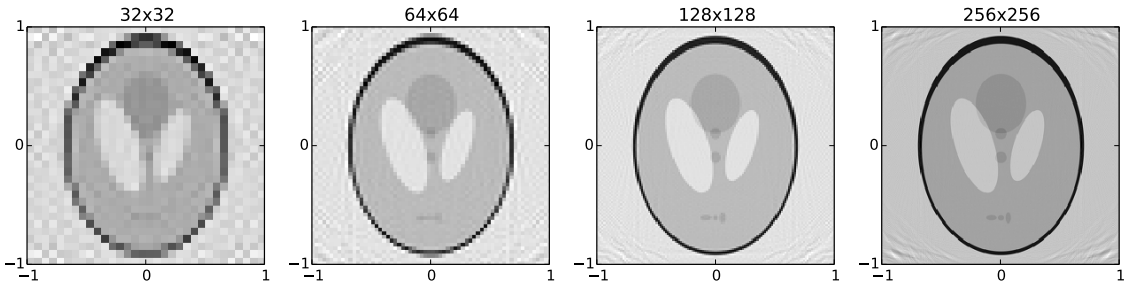


Abbildung 3: Tomographische Bilder, durch die entsprechende Dichtewerte ρ_{ij} bestimmt, für eine verschiedene Anzahl von Messungen und Strahlen pro Messung.

darauf, dass die von `take_measurement()` zurück gegebenen Daten an den korrekten Stellen in A eingefügt werden.

Aufgabe 3.2: Rekonstruktion des Bildes (2 Punkte)

Implementieren Sie die Funktion `compute_tomograph()` und rekonstruieren Sie mit Hilfe der Ausgleichsrechnung das tomographische Bild (falls Sie die Cholesky-Zerlegung nicht selbst implementiert haben, können Sie auf die `NumPy` Implementierung zurückgreifen). Eine Beispielrekonstruktion für 64 Messungen mit jeweils 64 Strahlen und einem 32×32 Gitter ist in Abb. **??** (ganz links) gezeigt.

Hinweis: Auch wenn die Matrix $A^T A$ aus mathematischer Sicht positiv-definit ist, kann es auf Grund der Verwendung von Gleitkommazahlen passieren, dass dies numerisch nicht der Fall ist. Da eine Computer-Implementierung der Cholesky-Zerlegung nur auf numerisch positiv semi-definiten Matrizen das gewünschte Ergebnis liefert, behilft man sich in der Praxis damit, ein kleines (z.B. das 10fache der Maschienengenauigkeit) positives Vielfaches der Identitätsmatrix auf eine Eingangsmatrix zu addieren, wenn diese nicht numerisch positiv semi-definite ist.