Lehrstuhl für Systemsimulation

Sommersemester 2025

Prof. Dr. Harald Köstler, Frederik Hennig, Michael Zikeli

Algorithmik kontinuierlicher Systeme
Aufgabenblatt 3 — Gleichungssysteme am Computer lösen

- Zu diesem Übungsblatt steht im StudOn-Kurs ein Zip-Verzeichnis mit Material bereit. Laden Sie dieses herunter und entpacken Sie es, bevor Sie mit den Aufgaben beginnen.
- Zu jeder Aufgabe gehört ein Python-Modul, dessen Name im Aufgabentitel gelistet ist. Füllen Sie die darin enthaltenen Funktions-Stubs mit Ihren Lösungen und geben Sie die Dateien über StudOn ab. Es handelt sich hierbei um **Einzelabgaben**. Sie können Ihre abgegebenen Dateien beliebig oft aktualisieren – nur die letzte abgegebene Version wird gewertet. Bitte ändern Sie nicht die Namen der hochzuladenden Dateien!
- Das Material zum Übungsblatt enthält außerdem ein Jupyter Notebook zur interaktiven Entwicklung Ihrer Lösungen, sowie eine automatisierte Test-Suite. Jede Teilaufgabe wird anhand von einer Reihe an Tests automatisch bewertet. Die *öffentlichen* Testfälle können Sie entweder in dem mitgelieferten Notebook, oder durch das Skript `blatt**_tests.py` auf der Konsole ausführen. Denken Sie daran, dass das Bestehen der öffentlichen Tests keine Garantie für Korrektheit ist.

Auf diesem Aufgabenblatt geht es um direkte numerische Lösungsverfahren für lineare Gleichungssysteme, basierend auf den in der Vorlesung vorgestellten Matrixzerlegungen. Diese sollen mithilfe von `numpy` implementiert werden. Matrizen und Vektoren werden dabei durch zweidimensionale `numpy`-Arrays dargestellt (Vektoren sind dabei praktisch Matrizen mit nur einer Spalte).

Wichtiger Hinweis: Sie müssen die Verfahren in diesem Übungsblatt selbst implementieren und dürfen nicht einfach existierende Funktionen wie `numpy.linalg.qr` verwenden.

Aufgabe 1 — LR-Zerlegung (12 Punkte) `lu.py`

Die LR-Zerlegung ist ein Standardverfahren zur Lösung linearer Gleichungssysteme. Es entspricht einer Gauss-Elimination, bei der alle Zwischenschritte in Matrizen gespeichert werden. Im Quellcode dieser Aufgaben ist die Zerlegung durch den englischen Begriff *LU-decomposition* bezeichnet.

a) Zeilen vertauschen (1 Punkt) Schreiben Sie eine Funktion `swap_rows`, die eine Matrix und zwei Zeilenindizes übergeben bekommt und die die angegebenen Zeilen dieser Matrix vertauscht.

b) Zeilen subtrahieren (1 Punkt) Schreiben Sie eine Funktion `subtract_scaled`, die eine Matrix, zwei Zeilenindizes und einen Skalierungsfaktor übergeben bekommt und die von der ersten Zeile das Vielfache der zweiten Zeile subtrahiert.

c) Pivot-Element finden (1 Punkt) Schreiben Sie eine Funktion `pivot_index`, die eine Matrix und einen Spaltenindex übergeben bekommt und den Zeilenindex des betragsmäßig größten Elements dieser Spalte zurück gibt. Elemente oberhalb der Diagonale sollen dabei ignoriert werden, so dass der zurückgegebene Zeilenindex immer größer oder gleich dem gegebenen Spaltenindex ist.

d) LR-Zerlegung (4 Punkte) Schreiben Sie eine Funktion `lu_decompose`, die eine quadratische, nichtsinguläre Matrix A übergeben bekommt und ein Tupel aus drei Matrizen P, L und R zurückgibt, so dass gilt $A = PLR$, wobei P eine Permutationsmatrix, L eine untere Dreiecksmatrix mit Einsen auf der Diagonale und R eine obere Dreiecksmatrix ist. Verwenden Sie dafür die Funktionen aus den vorherigen Teilaufgaben.

e) Vorwärtssubstitution (1.5 Punkte) Schreiben Sie eine Funktion `forward_substitute`, die eine untere Dreiecksmatrix L und einen Vektor b übergeben bekommt und das Gleichungssystem $Ly = b$ nach y löst. Nutzen Sie dabei die besondere Struktur der Matrix L.

f) Rückwärtssubstitution (1.5 Punkte) Schreiben Sie eine Funktion `backward_substitute`, die eine obere Dreiecksmatrix R und einen Vektor y übergeben bekommt und das Gleichungssystem $Rx = y$ nach x löst. Nutzen Sie dabei die besondere Struktur der Matrix R.

g) Gleichungssysteme lösen (2 Punkte) Schreiben Sie eine Funktion `linsolve`, die eine quadratische, nichtsinguläre Matrix A und beliebig viele Vektoren $b_1, \ldots, b_n$ übergeben bekommt, und ein Tupel aus Lösungen $x_1, \ldots, x_n$ zurückgibt, so dass für jede rechte Seite b_k gilt: $Ax_k = b_k$. Verwenden Sie dazu die Funktionen der vorherigen Teilaufgaben.

Aufgabe 2 — QR-Zerlegung (8 Punkte) `qr.py`

In diser Aufgabe sollen Sie die QR-Zerlegung einer reelen Matrix mit Hilfe von Givens-Rotationen implementieren.

a) Givens-Rotation (3 Punkte) Schreiben Sie eine Funktion `givens_rotation`, die für eine gegebene Matrix A, sowie einem Zeilenindex i und einem Spaltenindex j, eine Givens-Rotationsmatrix J zurückgibt, so dass das Produkt aus J und A im Eintrag i, j den Wert 0 hat.[1]

b) QR-Zerlegung (3 Punkte) Schreiben Sie eine Funktion `qr_decompose`, die für eine gegebene Matrix A ein Tupel aus den Matrizen Q und R zurückgibt, so dass gilt $A = QR$, wobei Q eine orthogonale Matrix und R eine obere Dreiecksmatrix ist. Im Gegensatz zur LR-Zerlegung soll dieses Verfahren auch für nichtquadratische Matrizen mit $m > n$ funktionieren.

c) Rückwärtssubstitution (0 Punkte) Wenn Sie die Funktion `backward_substitute` nicht schon für die LR-Zerlegung implementiert haben, holen Sie das jetzt nach. Ansonsten können Sie ihre vorherige Lösung hier wiederverwenden.

d) Gleichungssysteme lösen (2 Punkte) Schreiben Sie wie in Aufgabe 1 eine Funktion `linsolve`, die eine quadratische, nichtsinguläre Matrix A und beliebig viele Vektoren $b_1, \ldots, b_n$ übergeben bekommt, und ein Tupel aus Lösungen $x_1, \ldots, x_n$ zurückgibt, so dass für jede rechte Seite b_k gilt: $Ax_k = b_k$. Verwenden Sie dazu die zuvor implementierte QR-Zerlegung und Rückwärtssubstitution.

[1] Bei der Berechnung des Vorzeichens einer Zahl beachten Sie bitte, das `np.sign(0) == 0` ergibt, wir aber per Konvention `sgn(0) = +1` fordern.