Praktikum zur Numerischen Mathematik im Sommersemester 2021
Prof. Dr. Nils Rosehr, Hochschule München, 26. Mai 2021

Praktikum 4: Lösen linearer Gleichungssysteme

Implementierung einer Bibliothek

Schreiben Sie eine Bibliothek `lib4`, in der Sie mindestens die folgenden Funktionen implementieren.

- `Rückwärtsauflösen(Ab)`
- `Tausche(Ab, i, j)`
- `Addiere(Ab, a, i, j)`
- `Skaliere(Ab, a, i)`
- `Gauß_Elimination(Ab, Pivotisierung=False, Äquilibrierung=False)`

Dabei sei `Ab` eine erweiterte Koeffizientenmatrix eines quadratischen linearen Gleichungssystems, d.h. es gibt eine Spalte mehr als Zeilen und die letzte Spalte `b = Ab[:, -1]` entspricht der rechten Seite der Gleichung. Die Koeffizientenmatrix `Ab` soll als `numpy`-Array übergeben werden. Dann lässt sich z.B. mit `Ab[1, :]` auf die Zeile mit Index 1 (also die zweite Zeile) zugreifen. Weitere Informationen finden sich bei moodle im Dokument zu `numpy` im Abschnitt zu Indizierung.
Die fünf angegebenen Funktionen sollen (1) einen Lösungsvektor durch „Rückwärtseinsetzen“ aus `Ab` zurück geben, bzw. (2) die i-te und j-te Zeile von `Ab` vertauschen, bzw. (3) das a-fache der i-ten Zeile auf die j-te addieren, bzw. (4) die i-te Zeile mit a multiplizieren und schließlich (5) das Verfahren der Gauß-Elimination durchführen. Dabei soll der Rückgabewert der letzten Funktion ein Paar `(x, det)` sein bestehend aus dem Lösungsvektor x (oder `None`, falls dieser nicht eindeutig existiert) und der Determinante `det` der Koeffizientenmatrix `A = Ab[:, :-1]`. Die optionalen Argumente `Pivotisierung` und `Äquilibrierung` legen fest, ob Spalten-Pivotisierung bzw. Äquilibrierung durchgeführt werden soll.
Berechnen Sie zum Test die Lösung x der erweiterten Matrix

$$Ab = \left(\begin{array}{ccc|c} 1 & 5 & 2 & 49 \\ 0 & 1 & 7 & 121 \\ 1 & 2 & 9 & 162 \end{array}\right),$$

und verifizieren Sie Ihr Ergebnis, indem Sie sich Zwischenergebnisse ausgeben lassen und das Produkt Ax mit `A.dot(x)` berechnen.

Untersuchungen zur Genauigkeit der Lösung

- Berechnen Sie die Lösung von

  $$Ab = \left(\begin{array}{ccc|c} 11 & 33 & 1 & 1 \\ 0.1 & 0.3 & 2 & 1 \\ 0 & 1 & -1 & -1 \end{array}\right)$$

  exakt per Hand sowie mit `Gauß_Elimination(Ab)` mit und ohne Pivotisierung bzw. Äquilibrierung. Machen Sie sich das Ergebnis klar.

- In `Ab_x_list.data` ist eine Liste von 30 Paaren `(Ab, x)` definiert, wobei jedes `Ab` eine etwas größere erweiterte Koeffizientenmatrix und `x` der zugehörige Lösungsvektor ist. Laden Sie diese wie in `vorgabe4.py` gezeigt, und bestimmen Sie für alle Paare den absoluten Fehler Ihrer Lösung mit allen vier Berechnungsvarianten (mit/ohne Pivotisierung/Äquilibrierung), indem Sie eine Tabelle ausgeben, und vergleichen Sie die Genauigkeiten. Welche Varianten liefern die besten Ergebnisse? Wie erklären Sie sich das?
  (*Hinweis:* Sind `x1, x2, x3, x4` die vier berechneten Lösungen von `Ab`, so bietet sich der quadratische Fehler `sum(np.abs(x - xi)**2))` als Maß für den absoluten Fehler an.)

Anwendung PageRank

Implementieren Sie in `lib4` die folgenden beiden Funktionen:

- `LikeMatrix(links)`
- `PageRank(L, d=0.85)`

Es soll der PageRank-Algorithmus von Larry Page, wie in der Vorlesung vorgestellt, implementiert werden. Dabei sind die zu bewertenden N Objekte (Internetseiten, Tweets, etc.) mit 0 bis $N-1$ indiziert und `links` ist eine Liste der Länge N von Listen, so dass `links[i]` die Liste der Indizes ist, auf die das i-te Objekt verlinkt; diese Indizes entsprechen, wie in der Vorlesung beschrieben, den Einträgen ungleich Null in der i-ten Spalte der Like-Matrix. Die Funktion `LikeMatrix(links)`, soll dann die zu `links` gehörige Like-Matrix berechnen und zurück geben, und `PageRank` soll für eine Like-Matrix den PageRank-Vektor berechnen.

Berechnen Sie für

```
[[4, 3], [3, 4], [3, 0], [2], [2, 3, 0]]
```

den PageRank-Vektor. Warum ergibt sich gerade diese Reihenfolge bezüglich der PageRank-Bewertung?
Wir wollen nun Flughäfen mit guter Anbindung finden. Importieren Sie dazu die drei Variablen in `routes` durch `from routes import *`. In `airports` findet sich dann eine Liste der 1005 Flughäfen mit den meisten Verbindungen (Quelle: openflights.org/data.html). Wieder soll dem i-ten Eintrag dieser Liste (startend bei `0`) der Index i zugeordnet werden. In den Listen `routes_from_to` und `routes_to_from` befindet sich an der Position mit Index i die Liste der von i angeflogenen Flughäfen, bzw. die Liste der Flughäfen, die i anfliegen. Berechnen Sie nun den PageRank der Flughäfen in `airports` für beide Listen, und finden Sie jeweils die 15 Flughäfen mit dem höchsten PageRank. Speichern Sie diese als eine Liste von Paaren in der Form `BesterAbflug = [(PageRank, Flughafen), (...), ...]` bzw. `BesteAnkunft = [...]` sortiert nach absteigendem PageRank. Sie können hierzu den `sorted`-Befehl benutzen wie im Beispielcode in `vorgabe4.py`.

Abgabe

- Laden Sie das Archiv `P4vorgabe.zip` von moodle herunter, entpacken Sie es, und testen Sie Ihre Programme, indem Sie `test4.py` im gleichen Verzeichnis mit `python` ausführen. Erhalten Sie `ERROR`, so entspricht Ihr Programm nicht der Spezifikation von oben. Erhalten Sie `FAIL`, so ist Ihr Programm zwar lauffähig, aber die berechneten Werte sind fehlerhaft.
- **Abgaben, bei denen der Test gar nicht durchläuft oder mit `ERROR`, sind ungültig**, `FAIL` führt nur zu Punktabzug.
- **Bitte füllen Sie die Datei `info4.md` aus, und vergessen Sie nicht die Quellenangabe.** Abgaben mit unvollständiger Datei `info4.md` können nicht gewertet werden.
- Komprimieren und bündeln Sie alle oben erzeugten oder geänderten Dateien, indem Sie ein ZIP-Archiv erstellen. Sollten Sie nicht wissen, wie das geht, konsultieren Sie dazu die Dokumentation Ihres Betriebssystems.
- Benennen Sie Ihr ZIP-Archiv `P4.zip`.
- Schreiben Sie eine Email an `rosehr@hm.edu` mit Betreff `Numerik Abgabe`, Dateianhang `P4.zip` und beliebigem sonstigen Inhalt. **Wichtig ist, dass Betreff und Dateiname des Anhangs stimmen!**

Praktikumstermine P4: keine; **Abgabetermin:** 15.06.