Schnittstellendokumentation aufgabe4.py

Johannes Hübers, Cedric Brügmann, PPI11

19. Januar 2020

1 Einleitung

Dies ist die Schnittstellendokumentation des Pythonprogramms `aufgabe4.py`. Das Modul enthält Methoden zum Lösen von überbestimmten linearen Gleichungssystemen und zum Approximieren einer hängenden Kette mithilfe von Exponentialfunktionen. Zur Visualisierung der Experimente dienen Funktionen, die die Kettenapproximation, die Konditionen relevanter Matrizen und die Norm des Residuums, das heißt den Fehler der Approximation, zeichnen.

In `aufgabe4.py` ist eine main-Funktion implementiert, die die Methoden demonstriert und Plots für die Präsentation erstellt. Hierzu werden drei Dateien als Datenreihen eingelesen, `originaldaten`, `daten2`, `daten3`. Die Anfangs- und Endpunkte `(x,y)` der geordneten Datenreihen werden als Endpunkte einer hängenden Kette interpretiert. Mithilfe der Methoden zum Lösen überbestimmter linearer Gleichungssysteme können dann Parameter $a, b, c, d \in \mathbb{R}$ ermittelt werden, damit die Gestalt der hängenden Kette durch $y = ae^{dx} + be^{-dx} + c$ angenähert werden kann.

2 aufgabe4.py

2.1 solve_least_squares

Input:

- `a` *(numpy.ndarray)*: $m \times n$-Matrix vollen Spaltenrangs ($m \geq n$)
- `b` *(numpy.ndarray)*: 1D-Vektor mit m Einträgen

Returns:

- *numpy.ndarray*: 1D-Vektor mit n Einträgen, Lösung x von $a \cdot x = b$
- *float*: Residuum $\|a \cdot x - b\|$

Diese Funktion approximiert mit der Methode der kleinsten Quadrate die Lösung eines überbestimmten linearen Gleichungssystems $a \cdot x = b$ für eine $m \times n$-Matrix vollen Spaltenrangs a ($m \geq n$) und einen Vektor b. Hierfür wird die QR-Zerlegung der Matrix berechnet. Wenn die Matrix nicht vollen Spaltenrangs ist, bricht das Programm ab.

2.2 get_conditions

Input:

- `a` *(numpy.ndarray)*: $m \times n$-Matrix ($m \geq n$) vollen Spaltenrangs

Returns:

- *float*: Kondition von a
- *float*: Kondition von $a^T \cdot a$

Diese Funktion gibt für eine nicht notwendigerweise quadratische $m \times n$-Matrix ($m \geq n$) a vollen Spaltenrangs die Konditionen von a und des Produkts $a^T \cdot a$ aus.

2.3 solve_r

Input:

- `r` *(numpy.ndarray)*: Obere $m \times n$-Dreiecksmatrix ($m \geq n$) vollen Spaltenrangs
- `z` *(numpy.ndarray)*: Vektor der Länge m

Returns:

- `x` *(numpy.ndarray)*: Lösungsvektor x von $r \cdot x = z$ der Länge n

Die Funktion löst für eine obere $m \times n$-Dreiecksmatrix $r, m \geq n$, vollen Spaltenrangs und einen Vektor z das lineare Gleichungssystem $r \cdot x = z$ und gibt die Lösung x zurück.

2.4 read_data_set

Input:

- `path` *(string)*: Dateipfad des Datensatz
- `selection` *(iterable (ints))*: Gibt eine Auswahl von Datenpunkten (Zeilen), die in eine Liste eingelesen werden sollen an

Returns:

- `chain` *iterable (tuples of ints)*: Liste von eingelesenen Datenpunkten

Diese Funktion liest für gegebenen Dateipfad eine Auswahl von Datenpunkten (x, y) aus einem Datensatz ein und gibt sie als Liste aus. Der Pfad der Datei muss als Parameter angegeben werden und auch die Auswahl, ein *iterable* von *ints*, wobei jede Zahl auf eine Zeile (erste Zeile entspricht Zahl 0) in der Datei hinweist, die eingelesen werden soll. Jedes Wertepaar (x, y) erhält in der Datei eine eigene Zeile und die Einträge x, y werden durch ein Komma getrennt.

2.5 approx_chain

Input:

- `chain` *(iterable)*: Iterable von Datenpunkten `(x, y)` *(tuple)* einer Kette
- `d` *(float)*:

Returns:

- `x` *(numpy.ndarray)*: 1D-Vektor mit n Einträgen, Lösung x von $a \cdot x = b$
- `residuum` *(float)*:

Die Methode gibt für gegebenen Datensatz, ein *iterable* von *tuples* (x, y), die Punkte einer Kette repräsentieren, und Lösungsparameter d die anderen Lösungsparameter a, b, c, mit denen die Kette beschrieben werden kann, und das Residuum der kleinste-Quadrate-Lösung, das heißt den Approximationsfehler

2.6 approx_d

Input:

- `chain` *(iterable)*: Sammlung von Datenpunkten `(x, y)` *(tuple)* einer Kette

Returns:

- `best_d` *(float)*: Approximation des Lösungsparameters d

Die Funktion findet im Intervall $[0.1, 0.5]$ eine Näherung an den zur Minimierung des Residuums bei der Kettenapproximation optimalen Lösungsparameter d. Der Datensatz sollte eine Liste von *tuples* mit 2 *float*-Einträgen x, y, die Punkte einer Kette repräsentieren, sein.

2.7 plot_residuum_for_d

Input:

- `chain_list` *(iterable)*: Sammlung von Datensätzen (Listen von Datenpunkten `(x, y)` *(tuples)*), für die die Lösung approximiert und das Residuum gezeichnet werden soll
- `labels=0` *(iterable (strings))*: Sammlung von Legendenbeschreibungen *(strings)* für die gegebenen Datensätze. Bei Standard `0` wird keine Legende erstellt

Diese Funktion berechnet und plottet das Residuum $\|Ax-b\|$ für Lösungsapproximationen mit dem kleinste-Quadrate-Verfahren der Kettenapproximation zu gegebenen Datensätzen (Ketten) über Werte von d zwischen 0.1 und 0.5.

2.8 plot_conditions_for_d

Input:

- `chain_list` *(iterable)*: Sammlung von Datensätzen (Listen von Datenpunkten `(x, y)` *(tuples)*), für die die Konditionen der zur Problemlösung genutzten Matrix A und des Produkts $A^T \cdot A$ berechnet werden sollen
- `labels=0` *(iterable (strings))*: Sammlung von *strings,* enthält die Legendenbeschreibungen der Plots zu den Datensätzen aus `chain_list`. Beim Standard `0` wird keine Legende erstellt

Diese Funktion berechnet und plottet die Konditionen der Matrizen A und $A^T \cdot A$ für die Lösungsnäherung mit dem kleinste-Quadrate-Verfahren der Kettenapproximation für gegebene Datensätze (Ketten) über Werte von d zwischen 0.1 und 0.5.

2.9 plot_multiple_approximations

Input:

- `chain_list` *(iterable)*: Sammlung von Datensätzen (Ketten), Listen von Datenpunkten (*tuples* `(x, y)`), für die jeweils das Problem gelöst werden soll. Die erste Datenreihe wird im Schaubild mit eingezeichnet
- `d_list` *(iterable)*: Sammlung von Listen, die für Index-korrespondierende Datensätze aus `chain_list` die zu betrachtenden Werte für d enthalten
- `labels=0` *(iterable (strings))*: Sammlung der Beschreibungen *(strings)* der einzelnen Plots für jeden Datensatz und jedes d, Beschreibungen erscheinen in Legende. Bei Standard `0` wird keine Legende erstellt
- `title="Kettenapproximationen"` *(string)*: Titel des Plots, erscheint über dem Schaubild, Standard ist "Kettenapproximationen"

Diese Funktion berechnet die Kettenapproximationen für verschiedene Datensätze und Parameter d und stellt sie und die erste in `chain_list` gegebene Datenreihe in einem Schaubild dar.

2.10 plot_datasets

Input:

- `ds_list` *(iterable)*: Enthält die zu plottenden Listen von Tupeln `(x,y)` (Datenpunkten)
- `labels=0` *(iterable (strings))*: Sammlung der Beschreibungen *(strings)* der einzelnen Plots für jeden Datensatz und jedes d, Beschreibungen erscheinen in Legende. Bei Standard `0` wird keine Legende erstellt

Die Funktion dient zum Plotten von Datenreihen, ohne Lösungen.

2.11 main

main-Funktion von `aufgabe4.py`, wird bei Ausführung des Moduls als Hauptprogramm ausgeführt. Demonstriert Methoden aus dem Modul. Hierbei werden Plots der Kettenapproximationen für verschiedene Lösungsparameter d und Datensätze `chain`, `chain2`, `chain3`, und Plots des Residuums und der Konditionen von A und $A^T \cdot A$ über d für die Datensätze produziert.

3 Datenreihen

Zum Programm gehören die drei Textdateien `originaldaten`, `daten2`, `daten3`: In diesen werden Punkte `(x,y)` von zu approximierenden Ketten gespeichert, auf jeden Punkt entfällt hierbei eine Zeile, die Einträge `x`, `y` werden dann innerhalb dieser Zeile durch ein Komma getrennt.

Die Dateien `daten2` und `daten3` sind Modifikationen von `originaldaten`: Bei `daten2` wurden x- und y-Werte je auf die nächste ganze Zahl gerundet, bei `daten3` wurden sie abgerundet.