Übungen zu Computergrundlagen WS 2016/2017

Übungsblatt 11: Python III

20. Januar 2017

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 27.01.2017, 11:00 Uhr**.
- Schickt die Lösungen bitte per Email an Euren Tutor:
  - Montag 11:30 – 13:00 Uhr: Julian Michalowsky (`jmichalowsky@icp.uni-stuttgart.de`)
  - Montag 14:00 – 15:30 Uhr: Frank Uhlig (`fuhlig@icp.uni-stuttgart.de`)
  - Dienstag 14:00 – 15:30 Uhr: Patrick Kreissl (`pkreissl@icp.uni-stuttgart.de`)
  - Dienstag 15:45 – 17:15 Uhr: Kai Szuttor (`kai@icp.uni-stuttgart.de`)
  - Donnerstag 09:45 – 11:15 Uhr: Frank Maier (`fmaier@icp.uni-stuttgart.de`)
  - Donnerstag 15:45 – 17:15 Uhr: Evangelos Tzaras (`etzaras@icp.uni-stuttgart.de`)

Aufgabe 11.1: Datenanalyse mit NumPy(6 Punkte)

Ziel dieser Aufgabe ist es Messdaten mit NumPy zu verarbeiten. Es sollen Mittelwerte und Standardabweichungen von Daten berechnet werden, die aus Dateien eingelesen werden. Die Dateien mit den Daten liegen unter `/group/cgl/2016/11`. Schreibe ein Python-Skript, welches folgende Aufgaben erfüllt.

- **11.1.1** Lade die Daten aus den Dateien `sim_data_dens_0.0001.txt`, `sim_data_dens_0.001.txt`, `sim_data_dens_0.01.txt` und `sim_data_dens_0.1.txt` in ein NumPy-Array. Die Daten in der ersten Spalte sind die Zeiten, an denen die Messpunkte augenommen wurden, in der 2. Spalte stehen die Energien, in der 3. Spalte der Druck und in der 4. Spalte steht eine weitere Observable. (2 Punkte)
- **11.1.2** Berechne für alle drei Observablen die Mittelwerte und Standardabweichungen. (1 Punkt)
- **11.1.3** Plotte den zeitlichen Verlauf dieser Daten mit Hilfe des `matplotlib.pyplot`-Paketes. Stelle in diesen Plots auch zusätzlich den Mittelwert als rote, horizontale Linie dar. Deute die Standardabweichung als grüne, ebenfalls horizontale Linien an, so dass die obere und untere Linie genau einmal die berechnete Standardabweichung auseinander liegen. (1 Punkt)

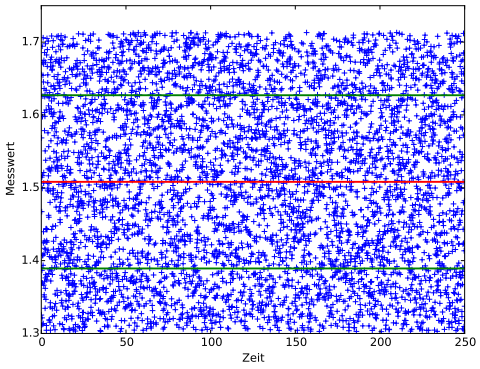

Abbildung 1: Beispielhafte Darstellung der Messdaten mit Mittelwert (rot) und Standardabweichung (grün).

- **11.1.4** Schaue dir für eine dieser Dateien die Histogramme der Observablen an. Erzeuge hierfür für jede der Observablen ein Histogramm der Messwerte mit Hilfe von `numpy.histogram()`. Plotte diese dann wieder mit der `matplotlib`. (2 Punkte)

Aufgabe 11.2: Fits in Python(4 Punkte)

Für den zweiten Teil dieses Aufgabenblatts soll die `curve_fit()`-Routine des `scipy`-Paketes verwendet werden. Die Daten, die gefittet werden sollen, befinden sich in der Datei `/group/cgl/2016/11/fit_data.txt`. Das Datenlayout ist `x y yerr`.

- **11.2.1** Lade die Daten wieder als NumPy-Array in Python und verwende die Funktion `curve_fit()` um eine Parabel $f(x) = ax^2 + bx + c$ an die Messdaten zu fitten. Beachte dabei, dass die Messdaten nicht alle auf einer Parabel liegen, stelle hierfür den Fit-Bereich entsprechend ein. (3 Punkte)
- **11.2.2** Plotte dann die "experimentellen" Daten zusammen mit der gefitteten Parabel. (1 Punkt)

Hinweise:

- Horizontale Linien können via `matplotlib.pyplot.axhline()` erzeugt werden.
- Die Grenzen der dargestellten Achsen können mit `matplotlib.pyplot.xlim()` und `matplotlib.pyplot.ylim()` eingestellt werden.
- `numpy.histogram()` gibt 2 NumPy-Arrays zurück. In dem ersten stehen die Anzahl der Treffer und in dem Zweiten die Ränder der bins.
- Um `curve_fit()` verwenden zu können muss `from scipy.optimize import curve_fit` am Beginn des Skriptes stehen.
- `curve_fit()` gibt 2 Arrays zurück. Im ersten stehen die Werte für die Fitparameter und in dem Zweiten die Kovarianzmatrix, die Imformationen über die statistischen Fehler für die Fitparamter enthält.
- Für `curve_fit()` muss eine Funktion definiert werden für ein Polynom mit 3 Parametern. Die Definition könnte zum Beispiel `def fit_func(x, a, b, c)` lauten.
- Um die Plot-Funktionalität der `matplotlib` Bibliothek verwenden zu können, kann mann alle nötigen Funktionen via `from matplotlib.pyplot import *` einbinden. Anstelle von z. B. `matplotlib.pyplot.plot(x, y)` muss man dann nur noch `plot(x, y)` schreiben. Das ist zwar bequem, bringt aber potentielle Probleme mit sich. Überlegt euch, warum diese Vorgehensweise problematisch sein kann und besprecht dies mit eurem Tutor! (*Keine* Aufgabe zum Abgeben!)
- Um Plots mit Fehlerbalken zu erzeugen, kann die Funktion `errorbar()` der `matplotlib` Bibliothek verwendet werden.