# Aufgabe: Lineare Regression aus Daten

Eine typische Situation aus dem Praktikum: Ihr habt für einen Versuchsteil viele Daten aufgenommen und sollt nun einen linearen Fit anlegen. Eine der besten Wege ist die Nutzung eines Python-Scriptes. Python ist eine recht einfache, aber mächtige, Programmiersprache, die mit vielen spezialisierten Paketen die Auswertung von Daten vereinfachen kann. Für eine gute und auf das Praktikum zugeschnittene Übersicht könnt ihr auf der Seite unseres Toolbox-Workshops nachschauen (https://toolbox.pep-dortmund.org/notes.html). Dort findet ihr auch noch einmal eine schöne Einführung, falls eure Kenntnisse bereits etwas eingerostet sind.

Für diesen Versuch habt ihr nun eine Reihe von Spannungen an eingetragen Linien gemessen und sollt daraus eine lineare Regression anfertigen, um die Parameter später weiterverwenden zu können.

Eure Werte sehen wie folgt aus:

| **Liniennummer** $N_{\mathrm{Linie}}$ | U / V |
|---|---|
| **1** | -19.5 |
| **2** | -16.1 |
| **3** | -12.4 |
| **4** | -9.6 |
| **5** | -6.2 |
| **6** | -2.4 |
| **7** | 1.2 |
| **8** | 5.1 |
| **9** | 8.3 |

Die Liniennummern $N_{\mathrm{Linie}}$ sollen dabei gemäß der Formel

$$D = (N_{\mathrm{Linie}} - 1) \cdot 6\ \mathrm{mm}$$

In die Abstände $D$ umgerechnet werden. Tragt dann $D$ gegen $U$ auf und führt eine lineare Regression durch.

Legt dafür bitte eine ausführbare Python-Datei an, die euch eure Datenpunkte und die Regression als .pdf-Datei speichert und eure Funktionenparameter mit Fehlern in der Konsole ausgibt.

## Tipps:

Doch wie liest man diese Werte am besten in Python ein? Gibt es vielleicht sogar einen Weg, wie euer Textsetzungsprogramm ebenfalls auf die gleiche Tabelle zugreifen kann?

Die Antwort sind .csv-Dateien. Dieses Format erlaubt es sowohl eurem Python-Script, als auch z.B. Latex auf die Daten zuzugreifen. Außerdem könnt ihr diese auch direkt während des Versuches nebenbei anlegen und spart euch das Kopieren später. Eine entsprechende Datei ist ebenfalls angehängt. Sie heißt „data.csv“. Dort findet ihr die oben dargestellte Tabelle wieder. Hier sind die einzelnen Spalten durch ein „ , “ getrennt. Dies ist wichtig um die Datei später richtig einlesen zu können. Die Spalten werden einfach durch eine neue Zeile dargestellt.

Jetzt sollten wir uns erstmal überlegen, was wir für Pakete benötigen:

- Numpy as np (Für das Einlesen und das Verarbeiten unserer Datenpunkte)
- matplotlib.pyplot as plt (Damit wir unsere Daten und die Regression auch darstellen können)
- from scipy.optimize import curve_fit (Für die lineare Regression)

Wie kann Python nun diese Tabelle einlesen? Dafür benötigt ihr das Paket „numpy“. Mit der „genfromtxt“-Methode könnt ihr dann zwei Variablen die Einträge in den Spalten zuweisen. Euer Funktionsaufruf sollte dann ähnlich aussehen:

```
linie, spannung = np.genfromtxt("data.csv",delimiter=",",unpack=True)
```

Das delimiter-Argument sagt Python nach welchem Zeichen ein Wert der nächsten Variable angehört.

Jetzt habt ihr eure Werte eingelesen. Allerdings könnt ihr mit den Werten noch nicht direkt etwas anfangen. Wie oben bereits erklärt sollt ihr die Liniennummern noch in Abstände umrechnen. Ist es vielleicht sogar sinnvoll sich eine neuer Variable „d“ dafür zu definieren?

Wenn ihr nun die Abstände berechnet habt, könnt ihr mit der Regression anfangen. Dafür benötigt ihr SciPy oder ein vergleichbares Paket. Eine Einführung zu SciPy findet ihr hier: https://toolbox.pep-dortmund.org/files/archive/2019/scientific-python.html . Für eine Regression müsst ihr euch nun zunächst eine Funktion definieren, die gefittet werden soll. In unserem Fall eine lineare Funktion. Überlegt euch also welche Parameter ihr bei der Definition benötigt und welche Funktion zurückgegeben werden muss. Danach könnt ihr mit der „curve_fit“-Methode die Regression vornehmen. Achtet hier darauf, dass ihr zwei Variablen zurückbekommt. Die Parameter und die Kovarianzmatrix. Letztere wird später für die Fehler benötigt. Alternativ könnt ihr auch die „polyfit“-Funktion aus dem Numpy-paket nutzen. Achtet dann darauf, dass ihr euch die Kovarianzmatrix mit zurückgeben lasst (Schlüsselwort cov=True).

Überlegt euch, wie ihr aus der Kovarianzmatrix die Fehler der Parameter erhaltet. (Wo stehen die in der Matrix und in welcher Form?) Die Kovarianzmatrix ist dabei als 2D-Array realisiert. Numpy besitzt bereits Funktionen um mit Matrizen rechnen zu können.

Lasst euch dann die Parameter mit entsprechenden Fehlern auf der Konsole ausgeben. Die Parameter liegen in einem Array vor(Wie greift man auf die entsprechenden Felder zu?)

Zum Schluss müsst ihr euer Ergebnis nur noch plotten. Dafür könnt ihr „matplotlib“ benutzen. Denkt daran:

- Messpunkte nicht verbinden
- Achsenbeschriftungen MIT Einheiten
- Legende
- Sonst auch vernünftige Bildunterschrift (ganze Sätze) => Erst in eurem Protokoll

Die Einführung zu „matplotlib“ könnt ihr hier finden: https://toolbox.pep-dortmund.org/files/archive/2019/matplotlib.html .