UNIVERSITÄT KONSTANZ
Fachbereich Physik (Theoretische Physik)
Dr. Stefan Gerlach
Raum P 817, Tel. (07531)88-3825
E-mail: stefan.gerlach@uni-konstanz.de

Universität Konstanz

Übungen zu Scientific Computing mit Python
Sommersemester 2025

Übungsblatt 6
Ausgabe 25.6., Übungen KW 28+29, Abgabe bis 18.7.

Monte-Carlo Integration, Statistik und Anpassung

1. **Aufgabe: Monte-Carlo-Integration**

   a) Wir wollen die Fläche eines Kreises mit Hilfe der Stein-Wurf-Methode bestimmen. Effektiv berechnen wir also das Integral

   $$4\int_0^1 \sqrt{1-x^2}\,\mathrm{d}x$$

   mit der Monte-Carlo-Integration.

   Schreibe dazu ein Python-Programm und bestimme N-mal zwei unabhängige Zufallszahlen a und b im Bereich $[0,1]$ und berechne das Verhältnis N_+/N, wobei N_+ die Versuche mit $a^2+b^2 \leq 1$ sind.

   Vergleiche das Ergebnis für verschiedene Werte von N. Wie gut konvergiert die Methode für $N \to \infty$ gegen den theoretischen Wert?

   b) **Einheitskugel**

   Die MC-Integration ist insbesondere bei hochdimensionalen Integralen sehr nützlich.

   Erweitere das Programm aus a), um das Volumen der d-dimensionalen Einheitskugel $V_d = \int \mathrm{d}^d x$ zu berechnen. Erzeuge dazu N-mal d unabhängige Zufallszahlen x_i im Bereich $[0,1]$ und berechne wieder das Verhältnis N_+/N, wobei N_+ hier die Versuche mit $\sum_{i=1}^d x_i^2 \leq 1$ sind.

   Stelle das berechnete Volumen für $N = 10^6$ abhängig von der Dimension $d = 1, .., 20$ dar und vergleichen Sie mit dem analytischen Ergebnis

   $$V_d = \frac{\pi^{d/2}}{\Gamma(1+d/2)}. \tag{1}$$

2. **Aufgabe: Zentraler Grenzwertsatz und Verteilungsfunktionen**

   (a) Lege eine Feld mit $N = 1000$ **Zufallsvariablen** X_i an, die von der Funktion `random.randint(1, 10)` generiert werden. Visualisiere das Ergebnis in einem **Histogram**. Welche (diskrete) Wahrscheinlichkeitsverteilung erhält man?

   (b) Betrachte mit $Y_k = \frac{1}{M}\sum_{i=1}^M X_i$ die **Summe von gleichverteilten Zufallszahlen**. Definiere dazu ein weiteres Feld der Länge N bestehend aus der Summe Y_k von jeweils $M = 100$ neuen Zufallsvariablen und stelle das Feld in einem weiteren Histogram dar.

(c) Plotte die Histogramme aus (a) und (b) nebeneinander mit plot.subplots und beschrifte die Achsen.

(d) Definiere eine Funktion die dir eine **Normalverteilung** mit Mittelwert μ und Standardabweichung σ berechnet. Plotte die Funktion für $\{\mu, \sigma\} = \{0, 1\}$ über einen x-Achsenabschnitt von -2 bis 2.

(e) Extrahiere den **Mittelwert** und die **Standardabweichung** des Histograms aus (b) und nutze diese in der Verteilungsfunktion aus (d) mit einem passenden x-Achsenabschnitt.

(f) Plotte das Ergebnis aus (e) zusammen mit dem Histogram aus (b) und vergleiche beide.

(g) Überlege dir, wie du aus der Dichtefunktion aus b) eine sinnvolle **Wahrscheinlichkeitsdichtefunktion** erhältst, sodass die Normalverteilung aus (e) dein Histogram aus (b) annähert.
Tipp: eine Wahrscheinlichkeitsdichtefunktion ist normiert.

3. **Aufgabe: (Nicht)lineare Anpassung**

Betrachte die drei Datenpunkte (-1,0), (0,2), (1,1).

(a) Schreibe ein Python-Programm, dass eine **Lineare Regression** des Datensatzes durchführt und plotte das Ergebnis.

(b) Finde mithilfe der **Polynomregression** das **Interpolierende Polynom** (2. Ordnung) und plotte es.

(c) Mit Hilfe von Gnuplot, LabPlot, Origin, usw. lassen sich Daten in einer GUI „fitten“. Finde anhand eines beliebigen Programmes und der obigen Daten heraus, wie das geht und teste es mit einer linearen Modellfunktion. Vergleiche die Ergebnisse mit denen aus a).

(d) Verwende SciPy (`scipy.optimize.curve_fit`) um die Daten mit einer linearen Funktion zu fitten. Welche Fitparameter liefert Python? Vergleiche die Ergebnisse mit denen aus a).