Institut für Theoretische Physik
der Universität zu Köln

Prof. Dr. Simon Trebst
Michael Becker, Peter Bröcker

Computerphysik

Übungsblatt 4

SS 2014

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2014-CompPhys.shtml
Abgabedatum: Montag, 5. Mai 2014 vor Beginn der Vorlesung

15. SciPy

Programmiertechniken

Python verfügt mittlerweile über eine riesige Menge an Modulen, die viele Funktionen für den tagtäglichen Gebrauch bereitstellen. Eines dieser Module ist **SciPy**, das einige der wichtigsten, für das wissenschaftliche Arbeiten relevanten Methoden implementiert.[1] In dieser Aufgabe konzentrieren wir uns zunächst auf das **Fitten** von gegebenen Daten – eine Prozedur, die in allen Bereichen der Physik von großer Bedeutung ist. Wir haben Ihnen auf der Homepage einen Datensatz bereitgestellt, der einer verrauschten Gaußfunktion entspricht, wie sie häufig experimentell gemessen wird. Um die relevanten Parameter, Mittelwert und Breite der Verteilung, zu bestimmen, verwenden wir die Methode der kleinsten Quadrate. In SciPy ist diese Funktion als `leastsq` im Modul `optimize` implementiert.

Wir nehmen also an, dass unsere Daten der Verteilung

$$y_i \stackrel{!}{=} C \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

folgen. Der Vergleich zwischen den gemessenen und den gefitteten Werten liefert uns die Residuen r_i:

$$r_i = \left| y_i - C \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right|$$

Diese gilt es nun zu minimieren. Wir definieren dazu eine Funktion `residuals`, die wir als Argument an `leastsq` übergeben werden:

```
from math import exp
from scipy.optimize import leastsq

def residuals(parameters, y, x):
    C, mu, sigma = parameters
    return y - C * exp( -(x - mu)**2 / (2 * sigma**2))
```

Nun müssen wir nur uns nur noch eine einigermaßen gute Vermutung für die Parameter (C, μ, σ) überlegen und erhalten dann die optimierte Version zurück:

[1] Tatsächlich sind im SciPy-Modul auch viele jener Funktionen vorhanden, die wir im weiteren Verlauf der Vorlesung + Übungen selber programmieren wollen.

```
p0 = [10., 2., 3.]

plsq = leastsq(residuals, p0, args=(y_data, x_data))
```

Vervollständigen Sie das auf der Kursseite verfügbare Programmskelett und plotten Sie dann sowohl die verrauschten Daten als auch den Fit in einen Plot.

In der Vorlesung haben Sie nun auch gelernt, wie man Integrale numerisch auswertet. Auf diesem Übungsblatt werden Sie die Trapez- und die Simpsonregel verwenden und miteinander vergleichen. Diese Verfahren werden standardmäßig verwendet und werden von SciPy durch das Paket `integrate` bereitgestellt. In den Aufgaben werden Sie den Definitionsbereich gleichmäßig diskretisieren. Eine Weiterentwicklung ist die Schrittweite der Diskretisierung an die Funktion anzupassen, also dort, wo sich die Funktion stark ändert und schwierig zu integrieren ist, mehr Gitterpunkte zu haben um den Fehler so zu verkleinern. Ein solch adaptives Verfahren ist mit der Funktion `quad` implementiert. Die Benutzung ist ähnlich einfach wie zuvor. Wir definieren zunächst eine Funktion die wir integrieren wollen und übergeben diese dann mit Definitionsbereich an `quad`. Als Rückgabewerte erhalten wir den Wert des Integrals sowie eine Fehlerabschätzung.

```
from math import cos
from scipy.integrate import quad

def intrgnd(x):
    return cos(x) ** 2 * x ** 5

result, error = quad(intgrnd, 0.1, 3.14)
```

Zum Abschluss sei noch einmal darauf hingewiesen, dass das SciPy-Modul deutlich mehr bietet als wir Ihnen hier präsentieren können. Im Verlauf der Vorlesung werden wir immer wieder auf einzelne Module zurückgreifen und sie vorstellen.

16. Rauschmittel

5 Punkte

In dieser Aufgabe wollen wir uns mit dem numerischen Ableiten von Daten beschäftigen.

1. Sie haben in der Vorlesung gelernt, dass die einfachste Methode um Daten numerisch abzuleiten ist, den zentralen Differenzenquotienten zu bilden:

$$f'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}, \tag{1}$$

   wobei wir $y_i \equiv f(x_i)$ definiert haben. Leiten Sie mit dieser Vorschrift die numerischen Daten ab, die Sie in der Datei **function1.dat** finden und plotten Sie die Ausgangsdaten und die berechnete Ableitung in einem Plot.

2. Leider sind (insbesondere experimentell gemessene) Daten in der Physik quasi immer mit einem Fehler behaftet. In der Datei **function2.dat** finden Sie die gleiche Funktion von vorher, allerdings mit einem kleinen Rauschen versehen. Berechnen Sie auch hier wieder die Ableitung und plotten das Ergebnis. Wie bewerten Sie die Brauchbarkeit ihres Ergebnisses? Erklären Sie, was das Problem ist.

3. Um die verrauschten Daten doch noch analysieren zu können, erinnern wir uns daran,

dass die Ableitung einer Funktion letztendlich nichts anderes als eine lokale lineare Approximation ist. Wir können also die Ableitung berechnen, indem wir an jeden Punkt eine Tangente anlegen. Diese berechnen wir mittels linearer Regression. Die Tangente an Punkt x_i ist gegeben durch

$$y = b_i \cdot x + a_i, \tag{2}$$

wobei hier die Steigung b_i schon die Ableitung am Punkt x_i ist. Diese Steigung b_i ergibt sich aus:

$$b_i = \frac{\sum_{j=i-n}^{i+n} \left(x_j - \bar{x}_i\right)\left(y_j - \bar{y}_i\right)}{\sum_{j=i-n}^{i+n} \left(x_j - \bar{x}_i\right)^2}.$$

Die Mittelwerte sind

$$\bar{x}_i = \frac{1}{2n+1} \sum_{j=i-n}^{i+n} x_j, \qquad \bar{y}_i = \frac{1}{2n+1} \sum_{j=i-n}^{i+n} y_j. \tag{3}$$

Berechnen Sie die Ableitung nun dreimal für $n = 5, 10, 20$ und plotten Sie alles in einem Plot. Welche Probleme können auftreten, wenn man n zu groß wählt?

Hinweis: Die Funktion, die den Daten zugrunde liegt, ist die Bessel-Funktion erster Art der Ordnung 0. Ihre Ableitung ist gegeben durch die negative Bessel-Funktion erster Art der Ordnung 1. Wenn Sie in Python das Modul "Scipy" importieren, sind diese beiden Funktionen als `scipy.special.j0(x)` bzw. `scipy.special.j1(x)` implementiert. Evtl. möchten Sie Ihre Ergebnisse damit vergleichen.

17. Simpson vs. Trapezregel

5 Punkte

Die Mathematik kennt eine Reihe von speziellen Funktionen, die gerade in der Physik eine besondere Relevanz entwickeln. Dazu gehören die **Kugelflächen-Funktionen**, welche in Problemen mit sphärischer oder zylindrischer Symmetrie auftreten. Vielleicht erinnern Sie sich an die Multipolentwicklung der Elektrostatik, oder aber Sie werden Ihnen in der Quantenmechanik bei der Lösung des Wasserstoffatoms wieder begegnen.

Sie ergeben sich als Lösungen der **Eigenwertgleichung** des Winkelanteils des Laplace-Operators:

$$\left(\frac{\partial^2}{\partial \vartheta^2} + \frac{\cos\vartheta}{\sin\vartheta}\frac{\partial}{\partial\vartheta} + \frac{1}{\sin^2\vartheta}\frac{\partial^2}{\partial\varphi^2}\right) Y_{\ell m}(\vartheta,\varphi) = -\ell(\ell+1) Y_{\ell m}(\vartheta,\varphi),$$

wobei wir uns hier auf den Spezialfall $m = 0$ beschränken wollen.

Die Lösung lautet dann

$$Y_{\ell 0}(\vartheta,\varphi) = \frac{1}{\sqrt{2\pi}} N_{\ell 0} P_{\ell 0}(\cos\vartheta)$$

mit den Normierungsfaktoren $N_{\ell 0} = \sqrt{\frac{2\ell+1}{2}}$ sowie den Legendrepolynomen

$$P_{\ell 0}(x) = P_\ell(x) = \frac{1}{2^\ell \, \ell!} \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell} \left(x^2 - 1\right)^\ell$$

Unser Ziel ist es nun, die **Orthogonalitätsrelation**

$$\int Y^*_{\ell 0}(\vartheta, \varphi) \, Y_{\ell' 0}(\vartheta, \varphi) d\Omega = \delta_{\ell \ell'}$$

der Kugelflächen-Funktionen durch numerische Integration für $\ell \in [0, \ldots, 3]$ zu überprüfen.

Die erste Methode, die Sie kennengelernt haben, ist die **Trapezregel**. Wie der Name schon suggeriert, wird der Flächeninhalt durch eine Reihe von Trapezen approximiert. Abhängig vom Verhalten der Funktion, dem Diskretisierungsschritt und der Größe des Integrationsbereichs kann dies bereits gute Ergebnisse liefern. Genauer ist allerdings die **Simpson Regel**, bei der die Funktion auch diskretisiert, aber dann intervallweise durch eine Parabel genähert wird.

Implementieren Sie ein Programm basierend auf der Beschreibung in der Vorlesung. Berechnen Sie die ersten vier Kugelflächen-Funktionen und führen Sie mit ihrem Programm die Integration aus (Was passiert mit der Integration über φ?). Verwenden Sie 15-20 Stützstellen für beide Verfahren. Beachten Sie, dass die Simpson-Regel zu gegebenen Stützstellen a und b immer auch $(a+b)/2$ hinzuzieht. Bewerten Sie, in welchen Fällen welche der beiden Methoden der anderen überlegen ist (bei gleicher Anzahl an Stützstellen).

18. Stau aus dem Nichts

optionale Aufgabe – 6 Punkte
Abgabe am Montag, 12. Mai

In dieser optionalen Aufgabe wollen wir die Entstehung von **Verkehrsstaus** anhand eines relativ simplen Modells verstehen, welches auf dem schon vorher erwähnten Konzept eines **zellulären Automaten** beruht – dem sogenannten **Nagel-Schreckenberg-Modell**, welches Anfang der 90er Jahre in Köln entwickelt wurde. Die Zellen sind dabei Fahrbahnabschnitte, die jeweils entweder von einem Fahrzeug belegt sind oder nicht. Jedes Fahrzeug hat zusätzlich zu seiner *Position* einen weiteren Parameter, nämlich eine *Geschwindigkeit*. Der Einfachheit halber umfassen die möglichen Werte für die Geschwindigkeit nur die Zahlen $0, 1, 2, 3, 4, 5$.

Man kann zu diesem Modell durchaus einen Bezug zur Realität herstellen, in dem man folgende Analogie verwendet:

$$\begin{aligned} \text{Länge einer Zelle} &\mathrel{\hat{=}} 7.5\text{m} \\ \text{Geschwindigkeit } i &\mathrel{\hat{=}} i \cdot 27\text{km/h} \\ \text{Iterationsschritt} &\mathrel{\hat{=}} 1\text{s} \end{aligned}$$

Der Iterationsschritt dieses zellulären Automaten besteht aus 4 Teilen:

1. **Beschleunigen**: Die Geschwindigkeiten aller Fahrzeuge werden um 1 erhöht, sofern Geschwindigkeit 5 noch nicht erreicht ist.

2. **Bremsen**: Bei allen Fahrzeugen, deren Abschnitt zum Vordermann (in Zellen) kleiner ist, als seine Geschwindigkeit, wird die Geschwindigkeit auf diesen Abstand reduziert.
3. **Trödeln**: Jedes Fahrzeug verringert seine Geschwindigkeit mit Wahrscheinlichkeit p um 1 (ohne erkennbaren Grund).
4. **Ausführen**: Jedes Fahrzeug wird um seine Geschwindigkeit (wiederum in Zellen pro Schritt interpretiert) nach vorne bewegt.

Beachten Sie dabei, dass alle Zellen *gleichzeitig* in den neuen Zustand übergehen.

Schreiben Sie ein Programm, welchen den oben beschriebenen zellulären Automaten für eine Fahrbahn von 200 Zellen mit *periodischen Randbedingungen* (wie eine Rennstrecke ohne Auf- und Abfahrt) umsetzt. Ermöglichen Sie dabei, als Parameter die *Trödelwahrscheinlichkeit* p und die *Verkehrsdichte* d (in Fahrzeugen pro Zelle) variabel einzustellen. *Visualisieren* Sie die Fahrzeuge (z.B. als farbig ausgefüllte Zelle) und zeigen Sie in jedem Schritt auch die Durchschnittsgeschwindigkeit aller Fahrzeuge an.

Wählen Sie zunächst eine gleichmäßig verteilte Startkonfiguration mit ruhenden Fahrzeugen und analysieren Sie die sich entwickelnde Verkehrssituation für folgende Parameter:

1. *Überfüllung* – $p = 0.0$, $d \in [0.2, 0.4, 0.6, 0.8]$,
2. *Sonntagsfahrer* – $d = 0.2$, $p \in [0.1, 0.2, 0.3]$,
3. *Ampelstart* – Stellen Sie alle Fahrzeuge zu Beginn direkt hintereinander auf und wählen Sie $p = 0.2, d = 0.2$. Wie lange dauert es, bis die entsprechende Durchschnittsgeschwindigkeit aus 2. erreicht ist?