Übungen zu Physik auf dem Computer SS 2012

Übungsblatt 12: Optimierung

11. Juli 2012

Allgemeine Hinweise

- Abgabetermin ist **Montag, 16.7.2012, 13 Uhr** (ausnahmsweise)!
- Zur Abgabe schickst Du die Lösungsdatei(en) im Anhang einer Email an Deinen Tutor:
  - Florian (`floh@icp.uni-stuttgart.de`; Dienstag, 15:45–17:15)
  - Dominic (`dominic@icp.uni-stuttgart.de`; Dienstag, 15:45–17:15)
  - Olaf (`olenz@icp.uni-stuttgart.de`; Mittwoch, 15:45–17:15)
- Die Übungen werden in Gruppen von jeweils zwei oder drei Leuten bearbeitet. Diese dürfen sich gerne von Blatt zu Blatt unterscheiden. Aus formalen Gründen muss allerdings jeder von Euch eine eigene Lösung abgeben. Schreibt bitte auf die Lösungen, mit wem Ihr zusammengearbeitet habt, um uns das Korrigieren zu erleichtern.
- Die Übungen finden statt im CIP-Pool des Instituts für Computerphysik (ICP) im Pfaffenwaldring 27.

Aufgabe 12.1 (5 Punkte): Steilster Abstieg mit Abstand

Man betrachte eine Verteilung von N Punkten im zweidimensionalen Raum $X = \{\vec{x}_i\}$. Für jedes Punktepaar $\vec{x}_i, \vec{x}_j$ kann man einen Abstand definieren und daraus eine „Abstandsmatrix“ $A_{ij} = A_{ji} = |\vec{x}_i - \vec{x}_j|$ erstellen. Der „Fehler“ der Verteilung ist dann wie folgt gegeben:

$$Q_A(X) = \sum_{i=0}^{N} \sum_{j=0}^{i} \left(|\vec{x}_i - \vec{x}_j| - A_{ij}\right)^2 \tag{1}$$

- 12.1.1 (1 Punkte) Implementiere eine Pythonfunktion `Q(X,A)`, die den Fehler wie in Gleichung (1) berechnet. Dabei sei X ein NumPy-Array der Länge $2N$, das alle Punktkoordinaten enthält. Dabei sollte es sich um einen eindimensionales Array der Länge $2N$ handeln! Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält.
- 12.1.2 (1 Punkt) Berechne den Gradienten $\nabla Q_A(X)$ der Fehlerfunktion. Gib die Rechnung im PDF-Format ab.
- 12.1.3 (1 Punkt) Implementiere eine Pythonfunktion `Qgrad(X,A)`, die den Gradienten der Fehlerfunktion berechnet. Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält.
- 12.1.4 (2 Punkte) Erstelle eine Matrix A_{ij} für die Berechnung eines gleichseitigen Dreiecks mit der Seitenlänge 100 und einen Satz aus 3 Punkten mit zufälligen Startkoordinaten. Verwende das Gradientenabstiegsverfahren mit Schrittweitensteuerung (`armijo_steepest_descent` aus dem Skript), um das gewollte Dreieck anzunähern. Erzeuge einen semilogarithmischen Plot (`semilogy`) des Fehlers $Q_A(X)$ über der Schrittzahl. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.

Aufgabe 12.2 (5 Punkte): Deutschlandkarte

Das Verzeichnis `/share/Courses/PC2012/12/` enthält die Dateien `demap.npy`, `demap.py` und `demap.png`. Kopiere Dir die Dateien. Das Pythonmodul `demap` definiert eine NumPy-(8×8)-Matrix `A` mit den Entfernungen zwischen den Städten Stuttgart, Frankfurt am Main, Köln, Hannover, Hamburg, Berlin, Dresden und München in Kilometer (in dieser Reihenfolge). Schau Dir das Modul am besten im Editor an.

Die Deutschlandkarte `demap.png` kannst Du mit Hilfe der folgenden Pythonbefehle darstellen:

```
image = matplotlib.pyplot.imread("demap.png")
matplotlib.pyplot.imshow(image)
```

- 12.2.1 (1 Punkt) Verwende die Pythonfunktionen aus Aufgabe 12.1, um die am besten zu dieser Matrix passende Punktverteilung zu berechnen. Erzeuge Plots wie in Aufgabe 12.1.4. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.
- 12.2.2 (2 Punkte) Die in Aufgabe 12.2.1 erzeugte Punktverteilung ist relativ zur Deutschlandkarte noch gedreht, nicht korrekt skaliert, und eventuell gespiegelt. Die Pythonfunktion `demap.adjust_to_map(X)` passt die $(N \times 2)$-Punktverteilung `X` an die Karte an. Erzeuge einen Plot, der die Karte `demap.png` und die berechnete und passend transformierte Punktverteilung zeigt. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.

  Hinweis Die Pythonfunktion `matplotlib.pyplot.annotate` kann dazu verwendet werden, um die einzelnen Punkte mit den Städtenamen zu beschriften. Die Städtenamen sind in der Pythonliste `demap.city_names` enthalten.
- 12.2.3 (2 Punkte) Schätze die Abstände (am besten, ohne dabei auf eine Karte zu gucken!) zwischen Stuttgart, Frankfurt am Main, Köln, Hannover, Hamburg, Berlin, Dresden und München in Kilometern und erzeuge eine NumPy-(8×8)-Matrix, die diese Abstände enthält. Wiederhole, was Du in Aufgabe 12.2.1 und 12.2.2 gemacht hast, mit Deiner geschätzten Matrix.

  Hinweis Die resultierende Karte kannst Du als Deine Vorstellung von Deutschland auffassen. Liegen einige der Städte in „Deiner“ Deutschlandkarte besonders weit weg von ihrer tatsächlichen Lage?