Übungen zu Physik auf dem Computer SS 2012

Übungsblatt 5: Signalverarbeitung und Datenanalyse

16. Mai 2012

Allgemeine Hinweise

- Abgabetermin ist **Montag, 14.5.2012, 13:00**
- 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 5.1 (6 Punkte): Radarsignale

Um mit Hilfe eines Radars die Entfernung verschiedener Objekte vom Radargerät zu bestimmen, wird ein wohldefinierter Impuls gesendet. Dieser wird vom Objekt reflektiert, und aus den Signallaufzeiten kann dann die Entfernung bestimmt werden.

Leider ist das reflektierte Signal beim Empfänger stark verrauscht, und es kommen auch Signale anderen Ursprungs beim Empfänger an. Daher wird die Kreuzkorrelation des gesendeten und des empfangenen Signals berechnet, um damit die Laufzeiten zu bestimmen.

Die Datei `/share/Courses/PC2012/05/radar.npy` enthält die (künstlich generierten) Daten eines Radarsignals. Kopiere die Datei in Dein Heimatverzeichnis.

Die Daten aus der Datei kannst Du in Python mittels des folgenden Befehls lesen:

```
t, signal1, ref1, signal2, ref2 = numpy.load('radar.npy')
```

Variable `t` enthält die Zeitachse des Signals, die Variablen `signal1` und `signal2` die gesendeten Signale. Die Variablen `ref1` und `ref2` enthalten das vom Empfänger empfangene verrauschte Signal.

- 5.1.1 (2 Punkte) Schreibe eine Pythonfunktion, die mit Hilfe der FFT-Funktionen von NumPy (`numpy.fft.rfft` und `numpy.fft.irfft`) die Kreuzkorrelation zwischen zwei beliebigen Reihen berechnet. Gib ein Pythonskript als Lösung ab, das die Funktion enthält.

- 5.1.2 (1 Punkt) Verwende die Pythonfunktion, um damit zunächst die Kreuzkorrelation der beiden Signale jeweils mit sich selbst zu berechnen. Gib ein Pythonskript als Lösung ab, das die Berechnung enthält. Erzeuge einen Plot der Kreuzkorrelationen über der Zeit und gib ihn im PDF-Format als Lösung ab.
- 5.1.3 (1 Punkt) Verwende die Pythonfunktion, um damit die Kreuzkorrelation der gesendeten mit den empfangenen Signalen zu berechnen. Gib ein Pythonskript als Lösung ab, das die Berechnung enthält. Erzeuge einen Plot der Kreuzkorrelationen über der Zeit und gib ihn im PDF-Format als Lösung ab.
- 5.1.4 (2 Punkte) Schreibe Antworten auf die folgenden Fragen in die Lösungsemail:
  - Welche Signallaufzeiten können sie aus den Kreuzkorrelationsplots ablesen? An wievielen Objekten wurde das Signal reflektiert?

    Hinweis Die Laufzeiten sind bei beiden empfangenen Signalen identisch.
  - Welchen Zweck erfüllt die Kreuzkorrelation des Signals mit sich selbst?
  - Die beiden Signale unterscheiden sich in der Länge des Impulses. Welchen Vor- bzw. Nachteil haben die beiden verschiedenen Signale?

Aufgabe 5.2 (4 Punkte): Datenanalyse einer Simulation

Die Datei `/share/Courses/PC2012/05/simulation.npy` enthält die Daten einer Molekulardynamiksimulation. Kopiere die Datei in Dein Heimatverzeichnis.

Es handelt sich dabei um die Daten der Simulation eines geladenen Kolloidteilchens (Ladung +300, Radius $50nm$) in einer Lösung aus einfach und mehrfach geladenen Ionen (Ladungen -1, $+1$ und -3)[1].

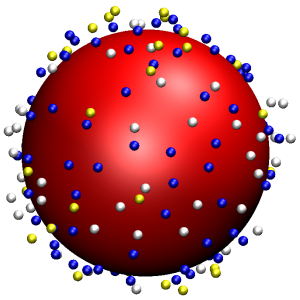

Abbildung 1: Geladenes Kolloidteilchen und Salzionen an seiner Oberfläche

Die Daten aus der Datei kannst Du in Python mittels des folgenden Befehls lesen:

```
t, n1, n2, n3 = numpy.load('simulation.npy')
```

Dabei ist `t` die Zeitachse und `n1`, `n2`, `n3` sind die Anzahlen der verschiedenen Ionentypen an der Oberfläche des Kolloids. Zu berechnen ist hier der Mittelwert dieser Messgrößen im Gleichgewichtszustand. Dabei ist zu beachten, daß die Simulation nicht im Gleichgewichtszustand gestartet wurde – zu Beginn der Simulation wurden die verschiedenen Ionen zufällig um das Kolloid herum verteilt. Am Anfang der Simulation muss sich also zunächst der physikalisch korrekte Gleichgewichtszustand einstellen („Equilibrierung“). Dies zeigt sich in der Messgröße daran, dass sie zunächst einen deutlichen Trend zeigt, bis sie nur noch um den gesuchten Mittelwert fluktuiert.

Um den gesuchten Mittelwert zu bestimmen, ist es also notwendig, festzustellen, wann die Simulation *equilibriert* ist, also die Messgröße keinen Trend mehr aufweist. Der Mittelwert sollte nur mit den Messwerten *nach* der Equilibrierung bestimt werden.

Leider gibt es kein formales Kriterium, stattdessen muß der Zeitpunkt der Equilibrierung visuell festgestellt werden. Da die Messgröße relativ stark fluktuiert, der Mittelwert allerdings etwas genauer sein sollte, als diese Fluktuationen, reicht ein einfacher Plot der Messgröße im Allgemeinen nicht aus.

Daher sollte die Zeitserie der Messdaten zur Bestimmung der Equilibrierung auf verschiedenen Skalen *geglättet* werden. Dazu ist es am einfachsten, die Messreihe mit einer Gaussfunktion der gewünschten Skala zu falten (die Skala ist dabei die Varianz σ der Gaussfunktion).

- 5.2.1 (1 Punkt) Schreibe eine Pythonfunktion, mit deren Hilfe zwei beliebige Datenreihen miteinander gefaltet werden können. Gib ein Pythonskript als Lösung ab, das die Funktion enthält.
- 5.2.2 (1 Punkt) Verwende die Funktion, um die Datenreihe der Messgröße `n2` auf den Skalen $100, 1000, 10000$ zu glätten und erzeuge Plots der ungeglätteten und geglätteten Messdaten. Gib die Plots im PDF-Format als Lösung ab.
- 5.2.3 (1 Punkt) Ab welcher Simulationszeit würdest Du die Simulation als equilibriert ansehen? Schreibe die Antwort in die Lösungsemail.
- 5.2.4 (1 Punkt) Schreibe ein Pythonskript um die Gleichgewichtsmittelwerte der drei Messgrößen zu berechnen. Gib das Pythonskript als Lösung ab, und schreibe die Mittelwerte in die Lösungsemail.

Literatur

[1] O. Lenz and C. Holm. Simulation of charge reversal in salty environments: Giant overcharging? *Eur. Phys. J. E*, 26:191–195, 2008.