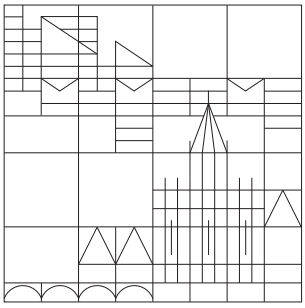
UNIVERSITÄT KONSTANZ
Fachbereich Physik
Dr. Stefan Gerlach (Theoretische Physik)
Raum P 817, Tel. (07531)88-3825
E-mail: stefan.gerlach@uni-konstanz.de



Übungen zur Einführung in die Computerphysik
Sommersemester 2008

Übungsblatt 4
Ausgabe 26.05.2008, Übungen 03.-05.06.2008 und 10.-12.06.2008, Abgabe bis 13.06.2008

C/C++ Programmierung
(`magnetpendel.c, Sudokus.tar.gz`)

11. **Aufgabe : Chaos I : Das Feigenbaum-Diagramm**
Ein einfaches Beispiel zur Veranschaulichung von chaotischem Verhalten ist die Abbildung

$$x_{n+1} = r x_n (1 - x_n).$$

Je nach Parameter r konvergiert die Abbildung bei unterschiedlichen Werten von x, oder auch nicht. Die sich so ergebenen "Fraktale" lassen sich in zahlreichen schönen Bildern darstellen.

(a) Schreibe ein C-Programm um das Konvergenzverhalten der Abbildung zu untersuchen, z.B. indem du die x_n für $n = 500..1000$ in Abhängigkeit von $r = 0..4$ ausgibst und alle Werte ein r-x-Diagramm darstellst. $x_0 = 0.5$ ist ein guter Startwert.

(b) Berechne den sog. Ljapunov-Exponenten, der ein Kriterium für chaotisches Verhalten darstellt mit Hilfe der Formel

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \ln(|f'(x_i)|).$$

(c) Stelle λ in Abhängigkeit von r dar und vergleiche das Bild mit dem Feigenbaum-Diagramm. Was lässt sich über das Konvergenzverhalten für $\lambda < 0$, $\lambda = 0$ und $\lambda > 0$ sagen?

12. **Aufgabe : Chaos II: Das Apfelmännchen**
Als nächstes wollen wir die Abbildung

$$z_{n+1} = z_n^2 + c$$

untersuchen. Wieder interessiert uns, für welche Werte von $c \in \mathbb{C}$ die Abbildung konvergiert. Der Bereich in der komplexen Ebene, in dem die Abbildung konvergiert, wird auch *Mandelbrot-Menge* genannt.

(a) Berechne mit dem Startwert $z = 0$ das n, ab wann $|z| > 10$ ist für alle Werte im Bereich $\mathrm{Re}\{c\} = [-2, 1]$ und $\mathrm{Im}\{c\} = [-1, 1]$. Stelle das Ergebnis grapisch mit gnuplot dar :

```
gnuplot> set view map
gnuplot> splot "mandelbrot.dat" w pm3d
```

oder, indem du direkt ein Bild erzeugst (siehe /usr/share/doc/packages/netpbm/doc/ppm.html) :

```
printf("P3\n# mandel.ppm\n400 600\n100\n");
printf("0 %d 0 ",pixel(a,b));
```

(b) Durch geeignete Wahl des Bereichs von $\mathrm{Re}\{c\}$ und $\mathrm{Im}\{c\}$ lassen sich verschiedene Ausschnitte im Detail berechnen. Versuche eine grundlegende Eigenschaft von Fraktalen, die *Selbstähnlichkeit*, durch Wiederfinden des Apfelmännchens in einem Ausschnitt zu bestätigen.

(c) Berechne alle Werte von $\mathrm{Re}\{z\}$ für $\mathrm{Im}\{c\} = 0$ und $n = 100 - 200$ im Bereich $\mathrm{Re}\{c\}$=[-2,0.25] und stelle $\mathrm{Re}\{z\}$ über $\mathrm{Re}\{c\}$ graphisch dar. Vergleiche das Bild mit dem Feigenbaum-Diagramm.

13. **Aufgabe : Chaos III : Das Magnetpendel**
Auch einfache physikalische Experimente führen häufig zu chaotischem Verhalten. Wir wollen das Magnetpendel, d.h. das Verhalten einer Stahlkugel an einem Pendel über drei Magneten untersuchen.

(a) Das Programm `magnetpendel.c` berechnet die Pendelbewegung und Gesamtenergie für einen festen Startpunkt. Versuche das Programm zu verstehen.

(b) Stelle die Pendelbewegung für verschiedene Startpunkte graphisch dar.

(c) Offensichtlich landet das Pendel am Ende immer über einem der drei Magneten. Modifiziere das Programm so, dass es für jeden Startpunkt im Intervall $x = [-1, 1]$ und $y = [-1, 1]$ den Endmagneten (1, 2 oder 3) ausgibt. Stelle das Ergebnis graphisch dar (vgl. Apfelmännchen-Aufgabe.)

(d) Wer möchte, kann das Programm auch für 2 oder 4 Magneten erweitern und sich die graphische Darstellung anschauen.

14. **Aufgabe(*) : Sudoku**
Wir wollen ein C++-Programm schreiben, dass uns die Arbeit abnimmt Sudokus zu lösen. Hier kurz die Regeln von Sudoku :

(a) Es soll eine 9x9 Matrix mit den Zahlen 1-9 gefüllt werden

(b) In jeder Zeile und Spalte, sowie in allen 9 3x3 Kästchen soll jede Zahl genau einmal vorkommen

Die Startprobleme lassen sich am besten als 9x9 Zahlenmatrix mit einer 0 für unbekannte Zahlen speichern. (siehe Beispiel-Sudokus).

(a) Schreibe ein "Hello World!"-Programm in C++

(b) Implementiere eine Klasse für eine Sudoku-9x9-Matrix mit alle notwendigen Methoden und Daten in einer eigenen Datei (z.B. `Matrix.cc`) und Header `Matrix.h`.

(c) Kompiliere das Programm aus den einzelnen Dateien mittels eines Makefiles.

(d) Schreibe die nötigen Methoden zur Ein- und Ausgabe der 9x9-Matrizen.

(e) Überlege dir einen Algorithmus um ein Sudoku-Puzzle zu lösen.

(f) Versuche, den Lösungsalgorithmus zu implementieren und an den Beispielen zu testen.