Institut für Theoretische Physik
der Universität zu Köln

Prof. Dr. Simon Trebst
Michael Becker, Peter Bröcker

Computerphysik

Übungsblatt 6

SS 2014

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2014-CompPhys.shtml
Abgabedatum: Montag, 19. Mai 2014 vor Beginn der Vorlesung

24. Visualisierung von Vektorfeldern *Programmiertechniken*

Die **Visualisierung** von **Kraftfeldern und Strömungen** mittels kleiner Pfeile kennen Sie bereits aus den Vorlesungen der ersten Semester. Bei gegebenen Vektorfeld sind diese auch mit python leicht zu erstellen. Wir beginnen damit, dass wir eine skalare Funktion ϕ diskretisiert auf einem zweidimensional Gitter gegeben haben, wie es zum Beispiel in der Aufgabe 25 der Fall sein wird. Um aus dem **Potential** das Kraftfeld zu berechnen, müssen wir den **Gradienten** berechnen. Dazu benutzen wir die numpy-Funktion *gradient*

```
# define X, Y domain for scalar function phi
values = np.linspace(-.1, .1, 20)
X, Y = plt.meshgrid(values, values)
# discretization step
dh = 0.01
# calculation of gradient
(grady, gradx) = np.gradient(phi, dh)
```

Beachten Sie, dass die Ordnung der Rückgabewerte etwas unintuitiv ist. Bei der Visualisierung des Vektorfeldes der Aufgabe 25 sollten Sie auf dieses Detail deshalb achten.

Um den Gradienten zu visualisieren, verwenden wir die Funktion quiver (dt. Köcher) aus matplotlib. Ihr übergeben wir wie auch schon bei den 3d Plots auf dem 0. Übungsblatt den Definitions- sowie Bildbereich:

```
plt.quiver(X, Y, gradx, grady)
```

Wir wollen nun das elektrische Feld einer elektrischen Ladung im Ursprung zu berechnen. Wir erinnern uns aus den KTP Vorlesungen, dass das Potential proportional zu r^{-1} ist, wobei r der Abstand vom Ursprung sei. Wir wollen also ein entsprechendes Potential *phi* im oben bereits definierten Bereich definieren und es dabei als numpy Array darstellen. Etwa indem wir es über

```
phi = np.empty((len(values), len(values)))
```

initialisieren und dann mit den entsprechenden Werten füllen.

Wer gleich zur Lösung springen möchte, der schaue sich das Beispiel vector_fields.py an.

25. Relaxen im Plattenkondensator *5 Punkte*

Den Plattenkondensator und sein **elektrisches Feld** haben Sie schon in der einführenden Vorlesung zur Experimentalphysik kennengelernt. Meist wird dabei die Annahme eines homogenen Feldes innerhalb des Kondensators gemacht, so dass die Feldlinien ein gerades Bündel innerhalb des Kondensators formen entsprechend der folgenden Abbildung.

Abbildung 1: Ein Plattenkondensator mit elektrischen Feldlinien

In dieser Aufgabe wollen wir überprüfen, in welchem Maße diese Annahme zutreffend ist. Dazu wollen wir das Feld des Plattenkondensators aus der Lösung der **Laplace-Gleichung**

$$\Delta\phi = 0$$

numerisch exakt berechnen, etwa indem wir auf die in der Vorlesung vorgestellte **Relaxationsmethode** zurückgreifen. Die Randbedingungen seien dabei so formuliert, dass die beiden Kondensatorplatten auf einem Potential von $+1$ bzw. -1 und der Rand der Fläche auf einem Potential von 0 liegt. Um die letzte Randbedingung zu rechtfertigen, müssen die Platten ausreichend weit vom Rand entfernt sein. In der Praxis können Sie die in der folgenden Abbildung dargestellte Konfiguration verwenden.

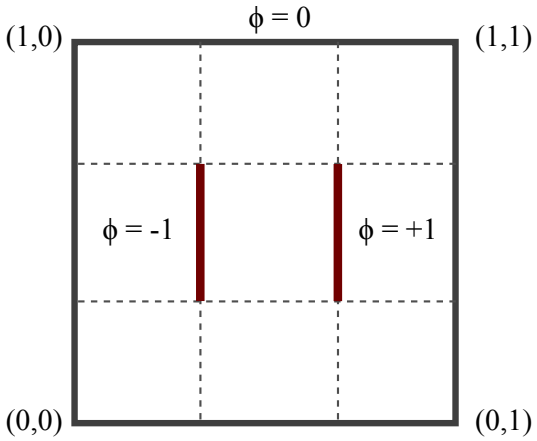

Abbildung 2: Startkonfiguration für die numerische Simulation

Implementieren Sie den in der Vorlesung vorgestellten Algorithmus für die Laplace-Gleichung mit den Randbedingungen aus Abbildung 2. Berechnen Sie dann das resultierende elektrische Feld und stellen Sie dieses dar, etwa wie in Aufgabe 24 beschrieben. Stimmt diese Feldkonfiguration mit der in Abbildung 1 überein? Ordnen Sie die Platten nun näher aneinander an. Wie ändert sich die Feldkonfiguration?

26. Heiße Neutronen – jetzt wird's kritisch *5 Punkte*

Numerische Simulationen spielen besonders in der Kernphysik bzw. Kerntechnik – aus naheliegenden Gründen – eine besonders zentrale Rolle. Als beispielhafte Anwendung aus diesem Bereich wollen wir die **Neutronenstreuung** in induzierten, nuklearen Spaltprozessen im Kühlbecken eines Kernreaktors simulieren.

Im Wesentlichen beruht der Mechanismus der **Kernspaltung** darauf, dass ein freies Neutron, welches auf einen Atomkern des Brennstoffs (etwa ^{235}U) trifft, eine Spaltung in zwei kleinere Kerne hervorruft und dabei 2 bis 3 neue Neutronen freisetzt

$$ {}^{1}_{0}\mathrm{n} + {}^{235}_{92}\mathrm{U} \longrightarrow {}^{141}_{56}\mathrm{Ba} + {}^{92}_{36}\mathrm{Kr} + 3\,{}^{1}_{0}\mathrm{n}\,. $$

Die freigesetzten Neutronen verursachen dann weitere Spaltungen aus, so dass eine Kettenreaktion ausgelöst wird, die es zu kontrollieren gilt.

In unserer Modellierung dieser Kettenreaktion konzentrieren wir uns auf die Neutronen, bzw. deren räumlich und zeitlich variierende Dichte, welche wir als $n(t,\vec{x})$ bezeichnen wollen. Die Entwicklung dieser Neutronendichte kann durch folgende partielle Differentialgleichung dargestellt werden

$$ \frac{\partial}{\partial t} n(t,\vec{x}) = D \cdot \Delta n(t,\vec{x}) + C \cdot n(t,\vec{x})\,. $$

Dabei setzen wir für die Diffusionskonstante $D = 11.3986332 m^2/s$ und für die effektive Neutronenerzeugungsrate $C = 1 s^{-1}$ an. Der Einfachheit halber wollen wir diese Prozesse in zwei räumlichen Dimensionen simulieren.

Um diese Differentialgleichung in zwei Raumdimensionen zu lösen, implementieren Sie einen Algorithmus mittels **finiter Differenzen** nach dem *Forward-Time-Centered-Space* Schema. Nehmen Sie dabei *Dirichlet-Randbedingungen* an, d.h. auf dem Rand ∂V des Systems gilt

$$ n(t,\partial V) = 0,\ \forall t\,. $$

Lassen Sie diesen Algorithmus für verschiedene Systemgrößen $L = L_x = L_y$ laufen. Wählen Sie dabei für den Reaktorkern ein konstantes Simulationsfeld von 100 x 100 ($N_x = N_y = N = 100$) Zellen, sodass sich die räumliche Diskretisierung zu $h = L/N$ berechnet. Das FTCS Schema verlangt für die Stabilität der Lösung, dass die zeitliche Diskretisierung $\tau \leq \frac{h^2}{2dD}$, wobei d die Raumdimension des Systems ist. Wählen Sie einen Disketrisierungsschritt von $\tau = \frac{h^2}{4dD}$.

Als Anfangsverteilung nehmen Sie einen Peak der Neutronendichte in der Mitte des Systems an, also

$$ n\,(t=0,(x,y)) = \begin{cases} 1/h^2 & \text{wenn } x = y = N/2 \\ 0 & \text{sonst.} \end{cases} $$

Simulieren Sie nun die zeitliche und räumliche Entwicklung der Neutronendichte für Reaktorkerne der Kantenlängen $L = 10, 15, 30$ – am besten in einer Animation – und beschreiben Sie Ihre Beobachtungen für die unterschiedlichen Parameter.

Tipp: Sie werden unter anderem beobachten, dass bei kurzen Laufzeiten Schachbrett-Oszillationen der Neutronendichte auftreten – warum? Diese können Sie glätten, indem Sie stets zwei Zeitentwicklungsschritte hintereinander durchführen und den Mittelwert der beiden als neue Konfiguration darstellen.

Optionale Zusatzaufgabe: Betrachten Sie einen Reaktorkern mit zusätzlichen **Moderatoren**, welche Neutronen absorbieren können. Zusätzlich zu den Dirichlet-Randbedingungen gibt es dann auch im Inneren Punkte, an denen $n(t,\vec{x}) = 0$ festgesetzt wird. Ordnen Sie die Moderatoren in einer octagonalen Konfiguration wie in der Abbildung an. Simulieren Sie nun die Neutronendichte für Systemgrößen $L = 20, 30, 75$.

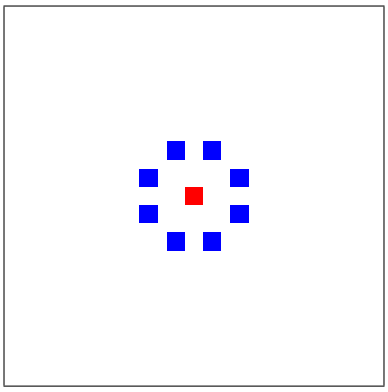

Abbildung 3: Anordnung für Moderatoren (blau) und Anfangsverteilung (rot).

Tipp: Sie können diese Anordnung auch direkt aus der **Bilddatei** in Ihr Programm importieren, indem Sie den Befehl

```
import scipy
neutrons = scipy.misc.imread('shieldedsource-8-small.png')
```

verwenden.

27. Otchaische Szillotienaon

(Chaotische Oszillationen)

optionale Aufgabe – 6 Punkte
Abgabe am Montag, 28. Mai

Anhand relativ einfacher Differentialgleichungen lassen sich eine Vielzahl **Dynamische Systeme** definieren, deren Zeitentwicklung allerdings hochgradig nicht-triviales Verhalten zeigen kann. Besonders spannend sind dabei **chaotische Phänomene**, also Zeitentwicklungen die sehr stark vom Ausgangszustand abhängen.

In dieser Aufgabe wollen wir zunächst ein besonders nützliches Werkzeug kennenlernen, mit dessen Hilfe wir derartige chaotisches Systeme sehr gut untersuchen können – sogenannte **Poincaré Schnitte**. Um einen derartigen Poincaré Schnitt zu erstellen, betrachten wir eine Ebene im **Phasenraum** des Systems und messen, wie oft diese durch eine Trajektorie bei gegebenen Anfangsbedingungen geschnitten wird. Besonders bei höherdimensionalen Systemen ist dies eine sehr hilfreiche Methode um einen Einblick in das Verhalten zu erlangen. Entscheidend ist allerdings, die richtige Ebene für das betrachtete Problem zu finden. Bei Systemen mit einer internen Periodizität – wie etwa dem getriebenen Pendel, welches wir unten

studieren werden – erweist es sich als gute Wahl, Ebenen entsprechend der periodischen Variable zu wählen.

Um ein Gefühl für das Erstellen von Poincaré Schnitten zu bekommen, wollen wir zunächst mit einem einfachen Beispiel beginnen – einem Pendel im Schwerefeld der Erde, dessen zeitliche Entwicklung durch die folgende Differentialgleichung gegeben ist

$$\ddot{\phi} = -\frac{g}{l}\sin(\phi)$$

Erstellen Sie ein Bild der Trajektorie im Phasenraum, aufgespannt durch $(\phi, \dot{\phi})$. Dazu können Sie entweder ihren eigenen Code von einem der vorherigen Übungsblätter nutzen oder das SciPy-Paket verwenden. Wie sehen für dieses System die Poincaré-Schnitte aus, wenn die Schnittebene in jedem Fall den Ursprung $(0,0)$ enthalten soll?

Wir fügen der Beschreibung des Pendels nun weitere Terme hinzu, welche die Zeitentwicklung chaotisch werden lassen. Das Pendel soll durch eine *Reibungskraft* abgebremst werden, die proportional zur Geschwindigkeit ist: $-\alpha\dot{\phi}$. Außerdem soll der *Aufhängepunkt* des Pendels mit der Periode T *oszillieren*. Dies führt zu einer weiteren Kraft, welche die folgende Form hat

$$\frac{A}{l}\left(\frac{2\pi\phi}{T}\right)^2 \cos\phi \cos\left(2\pi\frac{t}{T}\right)$$

Insgesamt erhalten wir also als Bewegungsgleichung

$$\ddot{\phi} = -\alpha\dot{\phi} - \frac{g}{l}\sin(\phi) + \frac{A}{l}\left(\frac{2\pi\phi}{T}\right)^2 \cos\phi \cos\left(2\pi\frac{t}{T}\right)$$

In diesem Fall ist die für den Schnitt relevante Variable t und wir speichern die aktuellen Koordinaten im Phasenraum bei Werten von $t = (a + n \cdot T)$, $n \in \mathbb{N}$. Für $\alpha = 0$, $A = 0$ reduziert sich das Problem auf das des reibungsfreien Pendels im Schwerefeld. Sie können nun untersuchen, wie sich das chaotische Verhalten manifestiert, wenn α auf einen festen Wert eingestellt wurde und A variiert wird. Werte für die sich chaotisches Verhalten sehr gut beobachten lässt sind

$$\alpha T = 0.2\pi, \quad T = 2\pi/3 \quad \text{und} \quad A = 2\,.$$

Sinnvollerweise setzen wir auch für dieses Problem $g = l = 1$. Gestartet werden kann die Simulation am Ursprung des Phasenraums, also bei

$$\phi(0) = 0 \quad \text{und} \quad \dot{\phi}(0) = 0$$

Achten Sie bei der Darstellung auch darauf, dass die Gleichungen explizit für den Fall eines Fadenpendels konstruiert wurden und der Wertebereich von ϕ damit auf $[-\pi, \pi]$ beschränkt ist.

Schließlich wollen wir noch ein zweites chaotisches System betrachten, den sogenannten **Duffing-Oszillator** – ein harmonischer Oszillator, welcher um einen Reibungsterm und einen kubischen Rückstellterm erweitert ist. Seine Zeitentwicklung lässt sich durch die folgende Differentialgleichung beschreiben

$$\ddot{x} = -\alpha\dot{x} - \beta x - \delta x^3 + \gamma\cos(\psi) \quad \text{und} \quad \psi = \omega \cdot t\,,$$

welche der obigen Differentialgleichung recht ähnlich ist und entsprechend einfach mit ihrem bisherigen Code simulieren lässt. Berechnen Sie auch hier Poincaré Schnitte im Phasenraum des Systems, welcher für ein gegebenes ψ von x und $\dot{x}$ aufgespannt wird. Untersuchen sie deren Abhängigkeit von den Parametern α, β, γ und δ.

Der auf der Kursseite gezeigte **Film** ist für

$$\alpha = 0.05, \quad \beta = 0, \quad \gamma = 7.5 \quad \text{und} \quad \delta = 1.0,$$

und eine kontinuierliche variierende Phase $\psi \in [0, 2\pi]$ gerechnet (wobei $\omega = 1$ gesetzt wurde). Als Anfangsbedingungen haben wir

$$x(0) = 3, \quad \dot{x}(0) = 4 \quad \text{und} \quad \psi(0) = 0$$

verwendet.

In beiden Modellen haben wir das bekannte mathematische Pendel mit nichtlinearen Termen erweitert. Dennoch zeigen sie sehr unterschiedliches Verhalten, das sich an den Poincaré-Schnitten sehr schön ablesen lässt. Der Duffing-Oszillator zeigt zwar chaotisches Verhalten, aber seine Trajektorie ist im Phasenraum räumlich begrenzt. Das getriebene Pendel zeigt hingegen Übergänge von $-\pi$ nach π, was bedeutet, dass das Pendel für eine Zeitlang auf dem Kopf pendelt. Dieses Verhalten lässt sich auch tatsächlich experimentell beobachten.

Filme erstellen

Zuletzt möchten wir noch erklären, wie Sie Filme wie die auf der Kurswebsite erstellen können. Wir gehen dabei davon aus, dass Sie den Wertebereich der periodischen Variable, das heißt t für das Fadenpendel bzw. ψ für den Duffingozillator diskretisiert haben und für jeden dieser diskrete Werte einen Poincaré-Schnitt (etwa in Form einer Matrix) vorliegen haben, welche sich etwa mit dem Befehl `imshow` darstellen lassen. Aus diesen Einzelbildern der einzelnen Poincaré-Schnitte wollen wir nun eine Sequenz und damit unseren Film zusammenstellen.

Damit das ganze schön aussieht, wollen wir eine Farbskalierung wählen. Zu diesem Zweck haben wir eine eigene Colormap erstellt, die Sie für das **Pendel** und für den **Duffing-Oszillator** von der Kursseite herunterladen können. Innerhalb des verlinkten Beispiel-Codes wird eine Matrix erzeugt und in einem sinnvollen Format abgespeichert, d.h. nummeriert mit einer festen Anzahl von Ziffern. Für das Generieren des Films benutzen wir das Programm `mencode`, dass mit dem Programm `MPlayer` ausgeliefert wird. Die meisten Linux-Distributionen haben `MPlayer` in den Standardpaketquellen. Windows und Mac OS X Versionen stehen auf der obigen Website zum Download bereit. Installieren Sie außerdem die `Binary Codec Packages`, die auf der Downloadseite etwa in der Mitte zu finden sind.

Nachdem Sie die Bilder generiert und `MPlayer` installiert haben, müssen Sie nur noch den Befehl

```
mencoder mf://movie_*.png -mf fps=20:type=png -ovc lavc -lavcopts
    vcodec=mpeg4:mbd=2:vpass=1 -oac copy -o movie.avi
```

in der Konsole ausführen womit die Bilder zu einem Film zusammengefügt werden. Die Anzahl von Bildern pro Sekunde und damit die Geschwindigkeit können Sie mit der Option `fps` kontrollieren.