Institut für Theoretische Physik
der Universität zu Köln

Prof. Dr. Simon Trebst
Michael Becker, Peter Bröcker

Computerphysik

Übungsblatt 4

SS 2013

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2013-CompPhys.html
Abgabedatum: Montag, 13. Mai 2013 vor Beginn der Vorlesung

In den einführenden Physik- und Mathematikvorlesungen haben Sie bereits verschiedene **Differentialgleichungen** und deren Lösung kennengelernt. Wie Sie aber sicherlich auch festgestellt haben, ist nicht jede dieser Gleichungen analytisch fassbar, bzw. überhaupt zu lösen. Wenn dann auch keine physikalisch gut motivierte Näherung als Lösungsansatz in Sicht ist, greifen wir gewöhnlich auf verschiedene numerischen Methoden zurück. In der Vorlesung wurden zwei der am häufigsten genutzten Verfahren – die Euler-Methode und das Runge-Kutta Verfahren – eingeführt. Diese wollen wir in den folgenden Übungen implementieren und anwenden.

16. Paralleles Programmieren mit MPI *Programmiertechniken*

Nicht nur moderne **Hochleistungsrechner**, sondern auch immer mehr Desktop- und Laptop-Rechner, sind in der Lage, Programme verteilt auf mehreren CPUs (oder CPU cores) auszuführen und ermöglichen so auch die Durchführung extrem aufwändiger Rechnungen.

Im folgenden wollen wir einen relativ einfachen Zugang kennenlernen, welcher es erlauben wird, ein gegebenes Programm für eine derartige Situation zu parallelisieren. Wir betrachten hierzu die Variante des sogenannten **Message Passing**, bei der jeder Prozess seinen eigenen Datenbereich besitzt, die einzelnen Prozesse aber miteinander kommunizieren und Daten austauschen können – ein Szenario, welches man als **distributed memory machine** bezeichnet. Das **Message Passing Interface (MPI)**, ist ein Standard, der eine Reihe von Funktionen definiert, die für die Kommunikation solcher Programme nützlich sind. Er wird zum Beispiel durch die Biblitotheken MPICH oder OpenMPI implementiert.

Um die Programme bei Ihnen zuhause zu testen und für weitere Aufgaben zu verwenden, sollten Sie zuerst MPICH (oder alternativ OpenMPI) und dann mit dieser Anleitung mpi4py, die Pythonschnittstelle, installieren. Sollten Sie unter Linux arbeiten, gibt es mpi4py auch in den verschiedenen Paketquellen und es wird z.B. OpenMPI als Abhängigkeit mitinstalliert. Im CIP-Pool ist bereits alles vorbereitet, so dass Sie direkt loslegen können ohne sich um die Installation zu kümmern.

Während wir für eine ausführliche Beschreibung des Pakets auf die Dokumentation verweisen wollen, beginnen wir hier direkt mit einem Beispiel:

```
1  # Importieren des MPI Pakets
2  from mpi4py import MPI
3
4  # Ihre Verbindung zur MPI-Welt
5  comm = MPI.COMM_WORLD
6
7  # Welche ID hat Ihr Prozess
8  rank = comm.Get_rank()
9
10 print "Hello World! I am process #", rank
```

Führen Sie dieses Programm mit dem Befehl

```
mpirun -np 4 python mpi_hello_world.py
```

aus. Was passiert? Der Befehl *mpirun* führt den Befel *python mpi_hello_world.py* vier mal aus. Die Anzahl der Prozesse wird über das flag *np* (=number of processes) spezifiziert. Außerdem stellt er die Funktionalität bereit, dass die Prozess untereinander kommunizieren und Daten austauschen können. Dazu bekommt jeder Prozess eine eindeutige ID zugeordnet, die in dem obigen Programm abgefragt und ausgegeben wird.

Die Kommunikationsfähigkeit wollen wir nun illustrieren. Laden Sie sich dazu das Programm mpi_send_recv.py herunter und starten Sie es mit verschiedenen Werten für die flag *np*. Was passiert?

Erläutern Sie, unter welchen Umständen sich mit diesen MPI-Techniken ein gegebenes Programm parallelisieren lässt. Wie würde man dabei vorgehen?

17. Euler Methoden

5 Punkte

In dieser Aufgabe werden Sie die einfachsten Methoden zur Lösung von gewöhnlichen Differentialgleichungen anwenden, um die Bewegung eines Pendels im Schwerefeld zu studieren. Konkret studieren wir also die Bewegungsgleichung

$$a(t) = \sin(x(t))$$

wobei $a(t)$ die Beschleunigung einer auf $m = 1$ gesetzten Masse sei. Beachten Sie, dass wir keine Kleinwinkelnäherung machen!

Implementieren Sie sowohl das forward- als auch das backward-Euler Verfahren und vergleichen Sie die Stabilität der Lösungen. Betrachten Sie dazu neben der Bahnkurve $x(t)$ auch die Energie und erklären Sie die Unterschiede.

18. Runge-Kutta Verfahren

5 Punkte

Das **Runge-Kutta Verfahren** haben Sie in der Vorlesung kennengelernt als eine Klasse von Methoden, die stabilere und präzisere Lösungen liefern können als die zuvor behandelten Euler-Methoden. In dieser Übung konzentrieren wir uns auf das vierstufige Runge-Kutta-Verfahren.

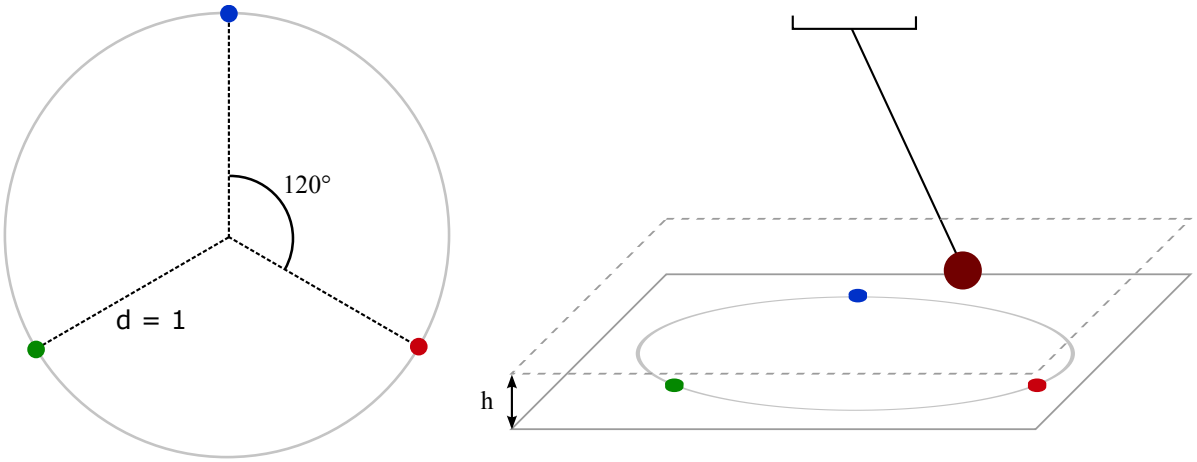
Wir wollen uns dabei zunächst graphisch klar machen, worin der Unterschied zwischen der Euler Methode und dem vierstufigen Runge-Kutta Verfahren liegt. Zur Erinnerung: Wir haben zum Ziel die Gleichung $\dot{y} = f(t,y)$ zu lösen, wobei wir $f(t,y)$ sowie die Anfangsbedingung (t_0, y_0) gegeben haben. Die Iterationsformeln geben Ihnen nur vor, wie sie vom Punkt (t_n, y_n) zum Punkt (t_{n+1}, y_{n+1}) kommen. Stellen Sie diesen Iterationsschritt in einem Koordinatensystem dar und beschriften Sie die einzelnen Teilschritte.

Implementieren Sie nun das vierstufige Runge-Kutta-Verfahren und nutzen Sie es um die Bewegungsgleichung des Pendels aus der vorherigen Aufgabe zu lösen.

19. Ein hart umkämpftes Pendel

optionale Aufgabe – 12 Punkte
Abgabe: Montag 27. Mai

Wir wollen nun die Situation aus den vorherigen Aufgaben etwas interessanter machen. Das **Pendel** sei nun magnetisch und schwinge in einer **Ebene**. Es wechselwirke mit dem **Schwerefeld** der Erde, sowie mit **drei Magneten** und werde durch **Reibung** gebremst. Die drei Magnete seien gleichmäßig auf einem Ring mit Radius $d = 1$ um den Ursprung angeordnet, wie auf der Skizze gezeigt.



Die gravitative Wechselwirkung nähern wir wie zuvor durch

$$\vec{F}_g \approx m\,g\,h\,\vec{e}_z \approx k_g \vec{r}\,.$$

Die Konstante k_g steuert die Stärke der Gravitation und $\vec{r}$ ist der aktuelle Ortsvektor des Pendels. Die Näherung ist, wie auch beim eindimensionalen Pendel, dass wir nur kleine Auslenkungen aus der Ruhelage betrachten.

Kommen wir nun zu den drei Magneten. Sie üben eine Kraft $\vec{F}_{k,i}$ auf das Pendel aus, wobei i der Index des Körpers $\{1,2,3\}$ ist und k_i eine weitere Kraftkonstante. So ergibt sich das folgende Kraftgesetz:

$$\vec{F}_{k,i} = k_i \frac{\vec{e}_r}{|\vec{r}^2|}\,.$$

Zuletzt fügen wir noch eine Reibungskraft $\vec{F}_r$ hinzu, die proportional zur Geschwindigkeit ist

und deren Stärke durch γ bestimmt wird

$$\vec{F}_r = -\gamma\vec{v}.$$

Lassen Sie sich nicht durch die Anzahl der Konstanten abschrecken. Die Simulation ist nicht viel schwerer als die vorherigen Aufgaben! Die Gesamtbewegungsgleichung ist also

$$\vec{a}(t) = \vec{F}_g + \sum_i \vec{F}_{k,i} + \vec{F}_r = k_g\vec{r} + \sum_i k_i \frac{\vec{e}_r}{|\vec{r}|^2} - \gamma\vec{v}, \tag{1}$$

wobei wir die Masse des Pendels $m = 1$ gesetzt haben und $a(t)$ die Beschleunigung bezeichnet. Wir nehmen an, dass die Auslenkung des Pendels in z-Richtung vernachlässigbar ist und betrachten dadurch effektiv ein zweidimensionales Problem, d.h. Sie berechnen nur die x und y Komponenten des Orts, der Geschwindigkeit und der Beschleunigung. Beachten Sie aber, dass die Ebene in der das Pendel schwingt im Abstand h über der Ebene der Magneten liegt. In der Berechnung der Abstandsvektoren $\vec{r}$ zwischen Pendel und Magneten geht der Abstand h also sehr wohl ein.

Die passende Wahl des Algorithmus zur Integration der obigen Bewegungsgleichung (1) sei Ihnen selbst überlassen. Der in der vorherigen Aufgabe behandelte Runge-Kutta-Algorithmus ist hierfür gut geeignet, aber Sie können auch einmal probieren, ob der backward-Euler Algorithmus stabil genug ist. Sie haben außerdem den **Verlet**-Algorithmus kennengelernt, der speziell für das Lösen von Bewegungsgleichungen entwickelt wurde.

Beginnen Sie damit, Trajektorien des Pendels zu berechnen, und stellen Sie diese grafisch dar. Hierzu müssen Sich sich insbesondere geeignete Abbruchbedingungen überlegen. Welche Endpositionen sind überhaupt möglich? In einem zweiten Schritt zeichnen Sie nun eine Karte, die jeder Startposition eine Farbe zuordnet, welche einer der möglichen Endpositionen entsprechen soll. Für die Magneten als Endposition können Sie z.B. die Farben so wählen wie in der Skizze.

Als gute Werte für die oben genannten Konstanten haben sich herausgestellt:

$$k_g = 0.2, \quad k_{k,1} = k_{k,2} = k_{k,3} = 20.0, \quad \gamma = 0.5, \quad h = 0.2$$

Für das Erstellen der Karte kann es sehr von Vorteil sein, MPI zu benutzen und das Programm auf mehreren CPU cores laufen zu lassen. Für eine Darstellung mit 800×800 Punkten und den genannten Parametern benötigt ein Intel Core i5 @ 2.5 GHz bereits etwas mehr als 40 Stunden Rechenzeit für unseren selbst geschriebenen Referenzcode. Mit MPI und einem Vierkernprozessor kann diese Zeit also deutlich verringert werden. Es sei schliesslich aber auch erwähnt, dass in der Forschung Laufzeiten von mehreren Wochen und sogar Monaten normal sind, um gute Ergebnisse zu bekommen – auch dann, wenn die verwendeten Codes so gut wie möglich optimiert sind, und die Rechnungen auf einem der grösseren parallelen Rechencluster laufen.