Partikelmethoden und gitterlose Diskretisierung

(Praktikum im Wintersemester 2011)

Leitung: Prof. Dr. M. Griebel
praktikum@ins.uni-bonn.de

Blatt 2

1 Das Lennard-Jones Potential

Bei einer Moleküldynamiksimulation sind in jedem Zeitschritt des Integrationsverfahrens zur Berechnung der Kräfte auf die Partikel die Wechselwirkungen der Partikel untereinander zu bestimmen und geeignet aufzusummieren. Exemplarisch wurde dies auf dem letzten Blatt mit einem skalierten Gravitationspotential demonstriert. Wir wenden uns nun idealisierten Molekülen zu, für die zwar die Gravitation zu vernachlässigen ist, aber verschiedene andere Wechselwirkung(stypen) relevant sind. Diese sind allerdings sind nicht trivial und erfordern eigentlich eine detailliert (oft quantenmechanische) Betrachtung, die aber nicht Bestandteil dieses Praktikums ist. Interessierte seinen an verwiesen und darin benannte, fortführende Texte.

Dementsprechend spricht man in der klassischen Moleküldynamik oft auch von Pseudopotentialen. Nichts desto trotz gibt es auch hier einige Feinheiten, die es bei der Behandlung zu beachten gilt, und diesen werden wir uns im Laufe dieses Praktikums zuwenden.

Betrachten wir als Beispiel das allgemeine Lennard-Jones Potential, welches für zwei Partikel im Abstand r_{ij} lautet

$$U(r_{ij}) = \alpha \cdot \varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^n - \left(\frac{\sigma}{r_{ij}} \right)^m \right], \qquad n > m. \tag{1}$$

Diese Potentialform findet vielfache Verwendung, z.Bsp. in abgewandelter Form zur Modellierung von Wasserstoffbrückenbindungen.

$$U(r_{ij}) = \frac{A}{r_{ij}^{12}} - \frac{B}{r_{ij}^{10}}. \tag{2}$$

An dieser Stelle wird unser Fokus aber auf dem klassischen Lennard-Jones Potential liegen, welches zur Beschreibung der Wechselwirkung in vielen verschiedenen Zusammenhängen wie zum Beispiel einfachen Gasen oder komplexeren biologischen Molekülen verwendet wird:

$$U(r_{ij}) = 4 \cdot \varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] = 4 \cdot \varepsilon \left(\frac{\sigma}{r_{ij}} \right)^6 \cdot \left[\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right]. \tag{3}$$

Dabei modelliert der anziehend wirkende Term die Van-der-Waals Wechselwirkung zwischen neutralen Atomen, während der zweite Term die Abstoßung der negativ geladenen Atomhüllen beschreibt.[1]

[1] Während der Exponent des Van-der-Waals Terms durch $-a \cdot r_{ij}^{-6}$ vorgegeben ist, ist die Wahl des abstoßenden Exponenten willkürlich und dient der schnelleren Berechenbarkeit.

Das Potential wird durch σ und ε parametrisiert. Der Wert ε gibt die Potentialtiefe an. Eine Erhöhung von ε führt somit zu festeren Bindungen. Der Wert σ gibt den Nulldurchgang des Potentials an.[2] Abbildung 1 zeigt das Lennard-Jones-Potential mit den Parameterwerten $\sigma = 1$ und $\varepsilon = 1$.

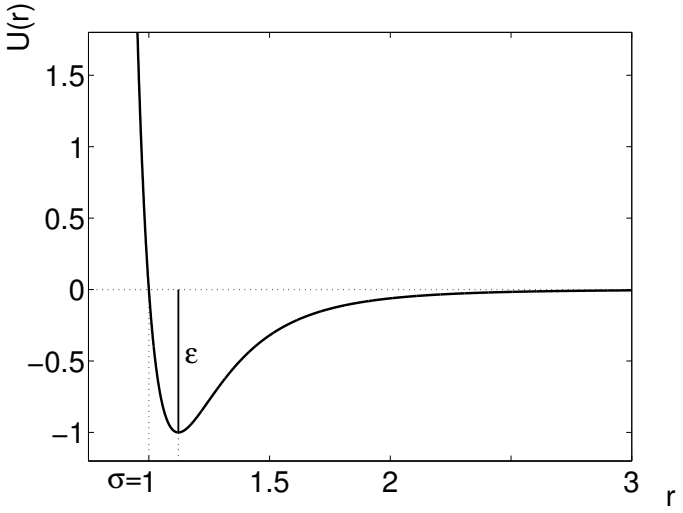

Abb. 1. Lennard-Jones Potential mit den Parameterwerten $\sigma = 1$ und $\varepsilon = 1$.

Mit $r_{ij} := \|\mathbf{x}_j - \mathbf{x}_i\|$ erhält man die Potentialfunktion für N Partikel dann als Doppelsumme zu[3]

$$\begin{aligned} V(\mathbf{x}_1, \ldots, \mathbf{x}_N) &= \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i<j}}^{N} U(r_{ij}) \\ &= 4 \cdot \varepsilon \sum_{j=1}^{N} \sum_{\substack{i=1 \\ i<j}}^{N} \left(\frac{\sigma}{r_{ij}}\right)^6 \cdot \left[\left(\frac{\sigma}{r_{ij}}\right)^6 - 1\right]. \end{aligned} \tag{4}$$

Die zugehörige Kraft $\mathbf{F}_i$ auf das Partikel i ergibt sich durch Gradientenbildung nach $\mathbf{x}_i$ zu

$$\begin{aligned} \mathbf{F}_i &= -\nabla_{\mathbf{x}_i} V(\mathbf{x}_1, \ldots, \mathbf{x}_N) \\ &= 24 \cdot \varepsilon \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{r_{ij}^2} \cdot \left(\frac{\sigma}{r_{ij}}\right)^6 \cdot \left(1 - 2 \cdot \left(\frac{\sigma}{r_{ij}}\right)^6\right) \mathbf{r}_{ij}. \end{aligned} \tag{5}$$

Hierbei ist $\mathbf{r}_{ij} := \mathbf{x}_j - \mathbf{x}_i$ der Richtungsvektor zwischen den Partikeln i und j an den Orten $\mathbf{x}_i$ beziehungsweise $\mathbf{x}_j$.[4] Die Kraft auf Partikel i besteht also aus einer Summe über die Kräfte $\mathbf{F}_{ij} := -\nabla_{\mathbf{x}_i} U(r_{ij})$ zwischen den Partikeln i und j,

$$\mathbf{F}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{F}_{ij}. \tag{6}$$

[2] Der Nulldurchgang der aus dem Lennard-Jones Potential resultierenden Kraft ist gegeben durch $2^{\frac{1}{6}}\sigma$, siehe auch (5).

[3] Sind die in (4) auftretenden Partikel nicht alle vom gleichen Typ, so können die Parameter ε und σ von den in der jeweiligen Wechselwirkung beteiligten Partikeltypen abhängig sein. Man schreibt dann ε_{ij} beziehungsweise σ_{ij}, um die Abhängigkeit von den Partikeln i und j deutlich zu machen.

[4] Man beachte wiederum, dass manchmal in der Literatur auch $\mathbf{r}_{ij} := \mathbf{x}_i - \mathbf{x}_j$ gewählt wird. Dann erhält man für die Kraft durch das Nachdifferenzieren von $\mathbf{r}_{ij}$ nach $\mathbf{x}_i$ das umgekehrte Vorzeichen.

Dabei verwenden wir für die Simulation zunächst einen globalen Satz von Parametern σ und ε.[5] Die Berechnung der Lennard-Jones Kraft zwischen zwei Partikeln soll analog zu der Gravitationskraft auf Blatt 1 implementiert werden.

Zu diesem Zweck soll eine neue Klasse `ljpotential` in den Dateien `ljpotential.hpp` und `ljpotential.cpp` implementiert werden.

2 Sourcecode verschönern

Da wir in diesem Praktikum immer auf den bestehenden Sourcecode aufbauen, empfiehlt es sich diesen jetzt zu optimieren und durchzudokumentieren. Folgende Hinweise sind zu beachten:

- Eine Zeile Kommentar pro Sinn-Code-Block, besser: Eine Zeile Kommentar pro Zeile Code.
- Beachtet dass die Dimension ab nun mit `DIM=3` festgelegt wird, testet ob Euer Code das verarbeiten kann. Das geht am einfachsten, indem ihr für die Aufgabe von Blatt1 entsprechend anpasst und testet.
- Korrekt einrücken, korrekt klammern. Sinnvoll klammern.
- Variablen sinnvoll benennen.
- *Sternchenaufgabe:* `<iterator>` einsetzen.
- *Sternchenaufgabe:* Komplexität der Kraftberechnung von $\mathcal{O}(N^2)$ auf $\frac{1}{2}\mathcal{O}(N^2)$ senken.

3 Erweiterung der Simulationsumgebung

Im Folgenden werden wir nur noch in einem begrenzten Gebiet arbeiten. Somit ist es notwendig die existierenden Klassen zu erweitern. Unser Gebiet ersteckt sich immer vom Nullpunkt ausgehend, es lässt sich folglich über die Länge in den verschiedenen Dimensionen definieren. Als Randbedingungen führen wir eine Randbedingung `BorderType` ein.

```
enum BorderType { unknown = 0, leaving = 1 };
```

Mit `leaving` wird dabei eine offener Rand bezeichnet. Ein Partikel der sich über die Grenzen des Gebiets hinaus bewegt wird aus der Simulation entfernt. Wenn der Typ `unknown` gesetzt ist, findet keine Randbehandlung statt (analog zu Blatt1).

Des weiteren soll der Code in Zukunft in der Lage sein, sowohl mit 2D als auch mit 3D Szenarien umgehen zu können. Dabei ist es völlig ausreichend, wenn die notwendige Festlegung zur Kompilierzeit getroffen wird, es ist explizit nicht notwendig diese Freiheit zur Laufzeit zur Verfügung zu Stellen.
Tipp: Bestehende "Schalter" müssen nicht ein zweites Mal eingebaut werden.

- `void world::read_Parameter( const std::string &filename )`
  Die Funktion muss so erweitert werden, dass aus der Parameter Datei muss nun auch die Größe des Gebiets und die Randbedingungen gelesen werden. Eine `.parameter` Datei wie in Beispiel 3.1 soll sich einlesen lassen.
  Achtung: Das Format der `.parameters` hat sich geändert (gegenüber der Variante auf dem ausgedruckten Übungsblatt)! Diese Variante ist leichter zu implementieren und kommt ohne das jonglieren mit strings aus[6].

[5] Wenn wir aber verschiedenartige Partikel behandeln, die unterschiedliche Parameter für das Lennard-Jones Potential erfordern, müssen wir, um die Symmetrie der Kraft $\mathbf{F}_{ij} + \mathbf{F}_{ji} = 0$ zu erhalten, so genannte Mischungsregeln für die individuellen Parameter verwenden.

[6] Wer das alte Format schon implementiert hatte, findet die alte Varinate (Expermiente und Übungsblatt hier`http://wissrech.ins.uni-bonn.de/teaching/praktika/md/ws2011/download/md-exp2.old.tgz`

- `void world::read_Particles( const std::string &filename )`
  Diese Funktion muss überarbeitet werden, so dass sie das vorgegebene `.particles` verarbeiten kann. Das Format ist wie folgt definiert:
  `id masse x0 x1 x2 v0 v1 v2`
- `void VelocityVerlet::handle_borders()`
  Es muss eine Funktion implementiert werden, welche für jeden Partikel überprüft ob er das Gebiet verlassen hat und entsprechend behandelt werden muss. In der Funktion `VelocityVerlet::timestep(real delta_t)` muss diese Funktion dann entsprechend aufgerufen werden. Sollten alle Partikel das Gebiet verlassen haben, soll die Simulation beendet werden.
- *Frage:* Welchen Einfluss haben offene Randbedingungen auf die Energieerhaltung?

Beispiel 3.1 `example.parameter`

```
name example2.1
delta_t 0.1
t_end 1.0
length 1.1 2.2 3.3
upper_border leaving leaving leaving
lower_border leaving leaving leaving
```

4 Visualisierung der Partikel

Um im Detail zu betrachten wie sich die Partikel verhalten, wollen wir diese visualisieren. Zu diesem Zweck verwenden wir das Programm `pymol`[7] Als Dateiformat wählen wir das einfache xyz-Format[8], in welchen einfach alle Zeitschritte hintereinander geschrieben werden. Für eine Simulation mit zwei Schritten und einen Partikel, der sich in Richtung `x[1]` bewegt, siehe Beispiel 4.1.

- Erweitere die Klasse `Observer` dahingehend, dass (analog zu `output_statistics()`) eine Datei mit dem Namen der Simulationsumgebung geöffnet wird und um die Funktion `output_xyz()` welche diese Datei in jedem Schritt mit dem aktuellen Konfiguration der Partikel beschreibt. Die Kommentarezeile (des xyz-Formats) sollte dabei den jeweiligen Zeitschritt enthalten.

  Sternchenaufgabe: Alternativ kann die entsprechenden Funktionen der Klasse `Observer` *als* `virtual` *deklarieren und eine neue Klasse* `ObserverXYZ` *von dieser erben und entsprechend erweitern.*
- *Hinweis* Das `xyz`-Format erwartet in der ersten Spalte eines Partikels stets das Atomsymbol. Wir betrachten zunächst nur Partikel und unterscheiden nicht den Typ, deswegen wählen wir als Atomsymbol stes das des einfachste Atoms `H`.
- Starte eine Simulation von einem sich bewegenden Partikel, öffne das resultierende `.xyz` mit `pymol` und mach Dich mit den Grundfunktionen des Programms vertraut.

[7] `http://www.pymol.org/`, ist im Praktikumsraum flaechendeckend installiert, im CIP Pool We6 verfügbar, zum herunterladen hier: `http://sourceforge.net/projects/pymol/files/Legacy/` unter Debian/Ubuntu `apt-get install pymol` mit Macports `port install pymol`.

[8] `http://en.wikipedia.org/wiki/XYZ_file_format`

Beispiel 4.1 `example.xyz`

```
1
Zeit: 0.00
H        0.000000        0.000000        0.000000
1
Zeit: 0.01
H        0.000000        0.000001        0.000000
```

5 Experiment: Partikel in der Energielandschaft

Wir wollen nun einige einfache experimentelle Untersuchungen mit unserem Programm durchführen. Dazu müssen einige Dateien unter `http://wissrech.ins.uni-bonn.de/teaching/praktika/md/ws2011/download/md-exp2.tgz` heruntergeladen und entpackt werden. Starte die Simulationen mit dem ersten Datensatz (`blatt2_example1.parameter` und `blatt2_example1.particles`), betrachte die erzeugten Ausgaben mit `pymol blatt2_example1.xyz`. Was lässt sich beobachten?

Verfahre analog mit dem zweiten Experiment (`blatt2_example2.parameter` und `blatt2_example2.particles`).

Beschreibe bzw. vergleiche das Verhalten der Partikel in den Experimenten. Woher rührt das unterschiedliche Verhalten?
Es empfiehlt sich diese Hypothese genauer zu untersuchen; erweitere zu diesem Zweck das dritte Simulationsexperiment (`blatt2_example3.parameter` und `blatt2_example3.particles`) um einen weiteren Partikel, so dass sich das bisherige Partikelverhalten nicht ändert und der neu Hinzugefügte sich genauso verhält.
Zeige nun theoretisch oder experimentell, dass dies auch mit vier Partikeln möglich ist.

6 Experiment: Elastischer Stoß

Wir wenden uns nun den zwei verbliebenen Experimenten (`blatt2_example4.parameter` und `blatt2_example4.particles` bzw. `blatt2_example5.parameter` und `blatt2_example5.particles`) zu. Diesmal sollten zusätzlich zu den Artikelbeobachtungen auch die Energien im System untersucht werden (z. Bsp. mit `gnuplot`). Beschreibe deine Beobachtungen und analysiere die beobachteten Effekte. Gibt es eine Möglichkeit die vorliegenden Experimente zu verbessern? Wenn ja, führe die notwendigen Änderungen durch.