

Partikelsimulation

Oliver Heidmann & Benjamin Warnke

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2017-01-12



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

informatik
die zukunft

Gliederung (Agenda)

- 1 Einleitung
- 2 Theorie
 - Allgemein
 - Verlet-Listen
 - Linked-Cell
 - Kombination
- 3 Meilensteine
- 4 Klassendiagramm
- 5 Output
- 6 Literatur

Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

3 Meilensteine

4 Klassendiagramm

5 Output

6 Literatur

Ziele

- Partikel-Simulation für kurzreichweitige Interaktionen
- optimiert & parallelisiert
- erweiterbar
- Cpp
- Auto-Tuning
- periodische Ränder

Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

3 Meilensteine

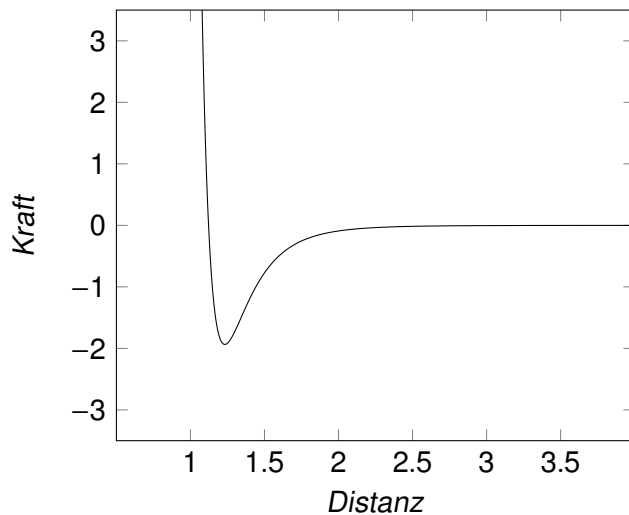
4 Klassendiagramm

5 Output

6 Literatur

$$f_{n,ij} = \left(\frac{48\epsilon_{ij}}{\sigma_{ij}^2} \right) \left[\left(\frac{\sigma_{ij}}{r_{n,ij}} \right)^{14} - \frac{1}{2} \left(\frac{\sigma_{ij}}{r_{n,ij}} \right)^8 \right] r_{n,ij}$$

Diagramm



Verlet-Algorithmus

- basiert auf dem Leapfrog-Verfahren
- eliminiert Ungenauigkeiten durch Integration
- verringert Abweichungen in den Resultaten zur Realität

Verlet-Algorithmus Formel

$$\vec{x}_{1,i} = \vec{x}_{0,i} + \vec{v}_{0,i}\Delta t + \frac{1}{2}\vec{a}_{0,i}\Delta t^2$$

$$\vec{x}_{n+1,i} = 2\vec{x}_{n,i} - \vec{x}_{n-1,i} + \vec{a}_{n,i}\Delta t^2$$

Resultierende Formel

$$A_{i,j} = 48\epsilon_{i,j}\sigma_{i,j}^{12}\Delta t^2$$

$$B_{i,j} = 24\epsilon_{i,j}\sigma_{i,j}^6\Delta t^2$$

$$\vec{x}_{n+1,i} = 2\vec{x}_{n,i} - \vec{x}_{n-1,i} + \sum_{j \in (p \setminus i)} \frac{A_{i,j} - B_{i,j}r_{n,i,j}^6}{r_{n,i,j}^{14}m_i} (\vec{x}_{n,j} - \vec{x}_{n,i})$$

Optimierungs Möglichkeiten

- kurzreichweitige Interaktionen
+ Zahlendarstellungsungenauigkeiten
→ cutoff-radius
→ Eliminierung von Berechnungen (kleiner Fehler)
- Parallelisierbar
 - OpenMP
 - CUDA
 - MPI

Verlet-Listen

- Nachbarschafts-Listen
- berücksichtigt fast nur relevante Partikel
- Neuaufbau der Listen alle x Iterationen
- Neuaufbau abhängig von momentaner maximalen Geschwindigkeit
- sehr geringe Cache-Kohärenz

Verlet-Listen






ListeFuer
Patikel 0

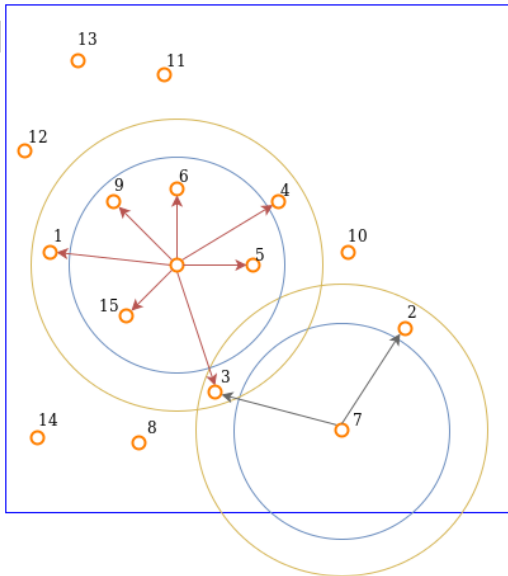
1	3	4	5	6	9	15
---	---	---	---	---	---	----

ListeFuer
Patikel 7

2	3
---	---

Legende

	Partikel
ist  Nachbar von	
#	Partikel Index
	$rc + dc$
	rc
	simulation border



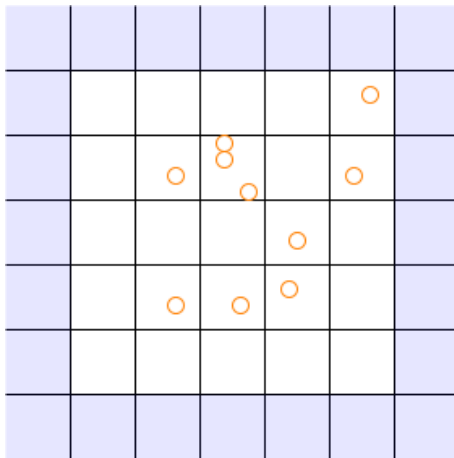
```
// cur = current    // idx = index    // part = particle
void example_iteration()
{
    build_lists_as_explained();

    //for loops over all parts through their indicies
    for(part_idx = 0; part_idx < part_count; part_idx++)
    {
        //loops over all neighbour idx entries
        for(list_idx = 0; list_idx < neighbour_list_size; list_idx++)
        {
            unsigned long cur_neighbour_idx = neighbour_list[list_idx];
            calculate_new_position(
                part_list[cur_part_idx],
                part_list[cur_neighbour_idx]
            );
        }
    }
}
```

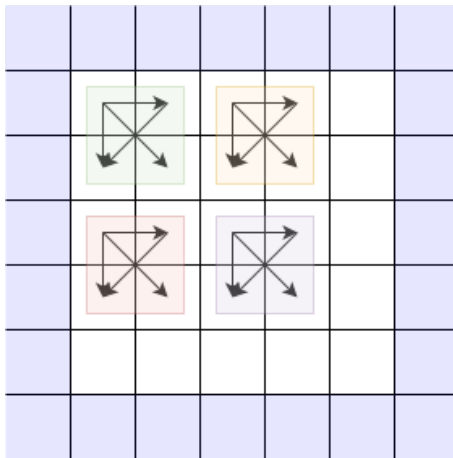
Linked-Cell

- räumliche Aufteilung des Raums
- direkte Umrechnung Position \leftrightarrow Zelle
- interagierende Partikel sind im Speicher nahe beieinander
→ Vektorisierung begünstigt
- Vorteil für MPI-Parallelisierung durch Randzellen

Partikel im Grid



Parallelisierung



Verlet-Listen & Linked-Cell

- Randübertretungen sind leichter zu erkennen
- Neuaufbau ist günstiger durch Zell-Struktur

Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

3 Meilensteine

4 Klassendiagramm

5 Output

6 Literatur

Hier stehen wir

- Planung
- Parameter zum Programmstart
- Laden (csv) & Generieren der Startdaten
- Implementation von Lennard-Jones mit dem Verlet-Algorithmus
- Datenausgabe (csv, avi)
- erste Simulationsdurchläufe
- Unit-Tests

TODOS

- Partikel sollen im Feld bleiben
- Kombination von Verlet-Listen & Linked-Cell
- Auto-Tuning
- Datenausgabe (verbreitete Formate & binär)
- prüfen der Energieerhaltung
- Parallelisierung

Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

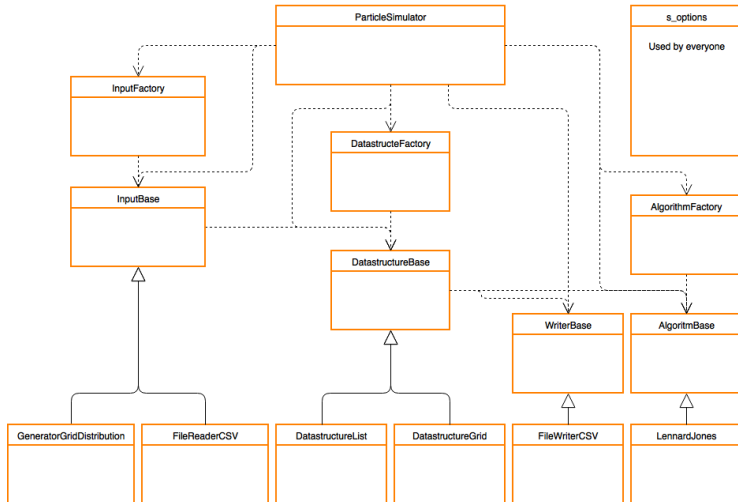
3 Meilensteine

4 Klassendiagramm

5 Output

6 Literatur

Klassendiagramm



Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

3 Meilensteine

4 Klassendiagramm

5 Output

6 Literatur

Output File

```
1 "ID","PositionX","PositionY","PositionZ"  
2 0,      1.5,      2.0,      2.0  
3 1,      3.5,      2.0,      2.0
```

Listing 1: data.0.csv

Parameter für das Program

- algorithm=LENNARD_JONES
- data_structure=GRID
- max_iterations=7090
- write_fequency=10
- cut_off_radius=2.5
- timestep=0.005
- bounds=4/4/4

Gliederung (Agenda)

1 Einleitung

2 Theorie

- Allgemein
- Verlet-Listen
- Linked-Cell
- Kombination

3 Meilensteine

4 Klassendiagramm

5 Output

6 Literatur

Literatur

- M-Griebel, S. Knapek, G. Zumbuschm, A. Caglar:
Numerische Simulation in der Moleküldynamic. Springer,
2003
- D.C Rapaport: The Art of Molecular Dynamics Simulation -
2nd edition, Cambridge University Press, 2004