N-Body Simulation

Uwe Ehmann, Hanna Ketterle

26.03.2015

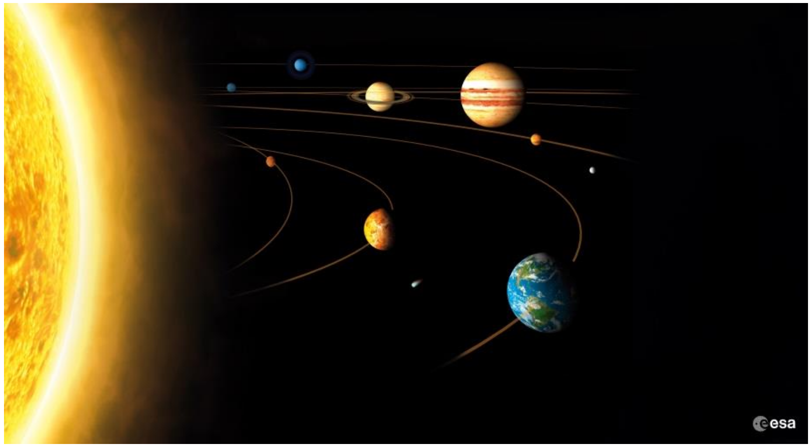
Agenda

1. Background
2. CUDA Implementation
3. CUDA/OpenGL Interoperability
4. OpenGL
5. Gravity vs. MD simulation
6. Test Scenarios

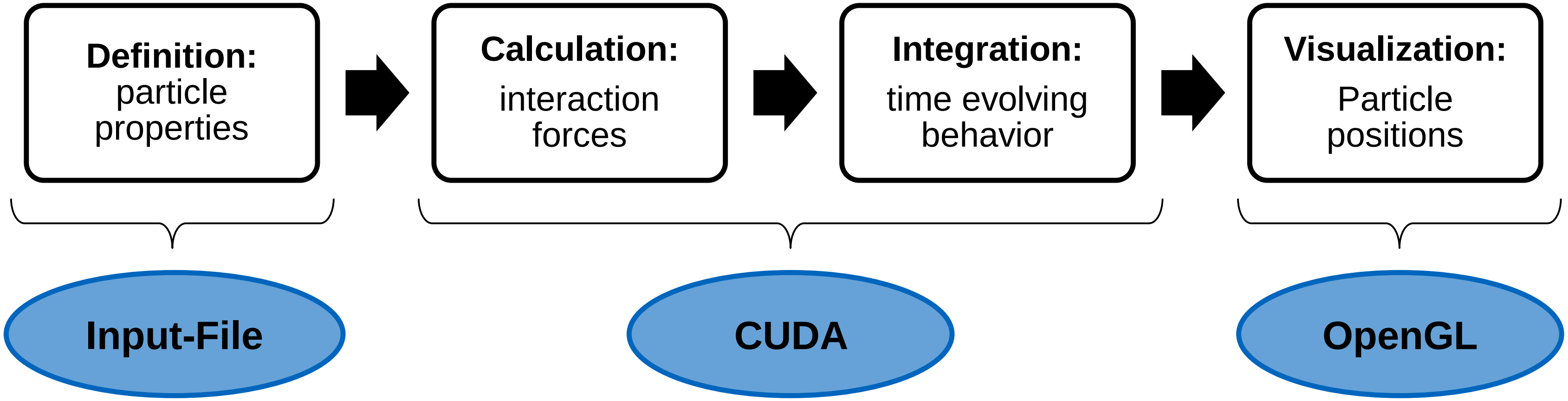
Background

What is an N-Body system?

- System of N particles
- Particles interact with each other
- Example: solar system



- Tasks:



Definition: particle properties

Calculation: interaction forces

Integration: time evolving behavior

Visualization: Particle positions

Input-File

CUDA

OpenGL

Background

Force Calculation:

- Dependent on distance between particles
- Total force = sum of all interaction forces
- $F_{tot} = \sum_{i=1}^{n} F_i$ → for each particle

Motion:

- Newton formula: $F = m * a(t) = m * \ddot{x}(t)$

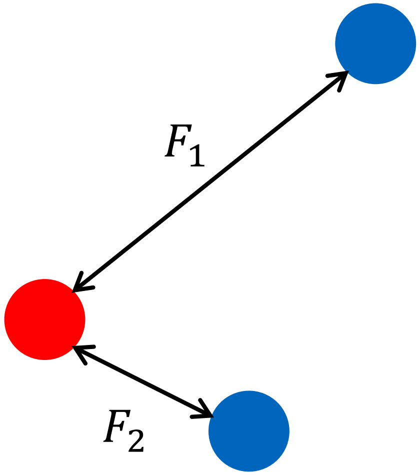
Integration:

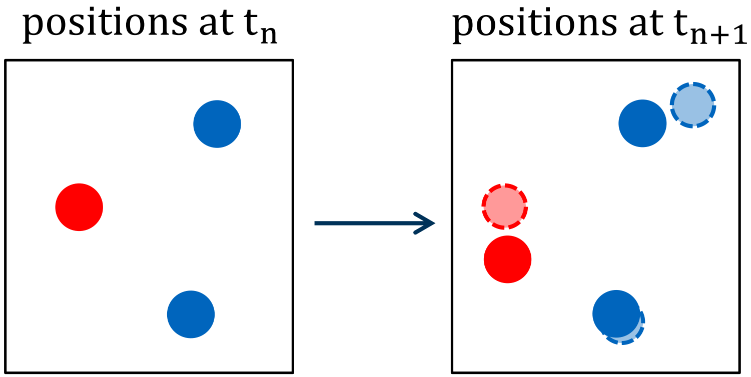
- Euler Integration:

$$x_{n+1} = x_n + v_n \cdot \Delta t + O(\Delta t^2)$$
$$v_{n+1} = v_n + a_n \cdot \Delta t + O(\Delta t^2)$$

- Velocity-Verlet Integration:

$$x_{n+1} = x_n + v_n \cdot \Delta t + \frac{1}{2} \cdot a_n \cdot \Delta t^2 + O(\Delta t^4)$$
$$v_{n+1} = v_n + \frac{(a_n + a_{n+1})}{2} \cdot \Delta t + O(\Delta t^2)$$





CUDA Implementation

Main Calculations with CUDA:

- Force calculation
- Integration

→Every thread calculates force on one particle

Special features of implementation:

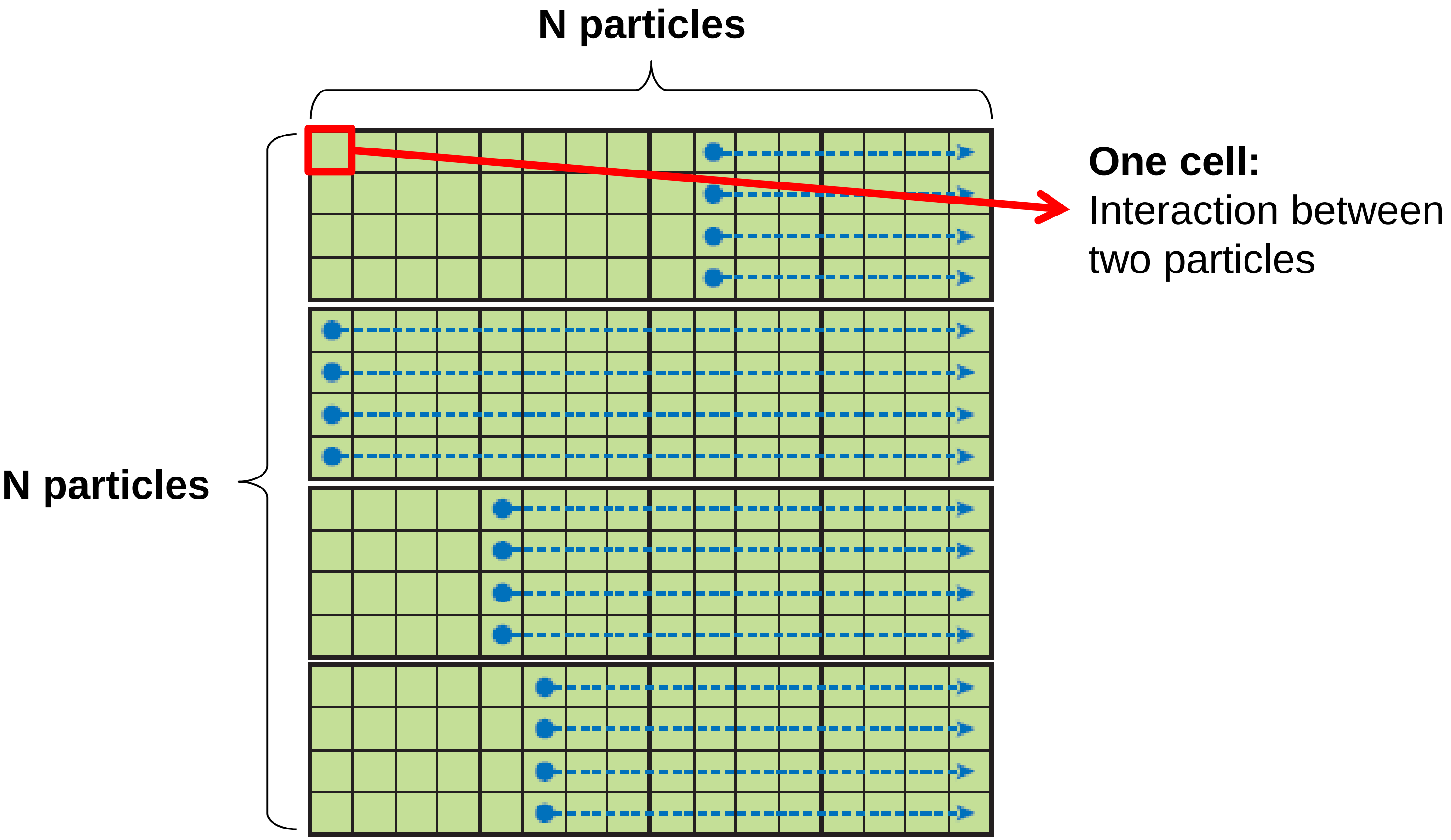
- Usage of shared memory
- Selection between different force-models possible
- Selection between different integrators possible

Used algorithm:

fast N-Body simulation (Nyland et. Al)

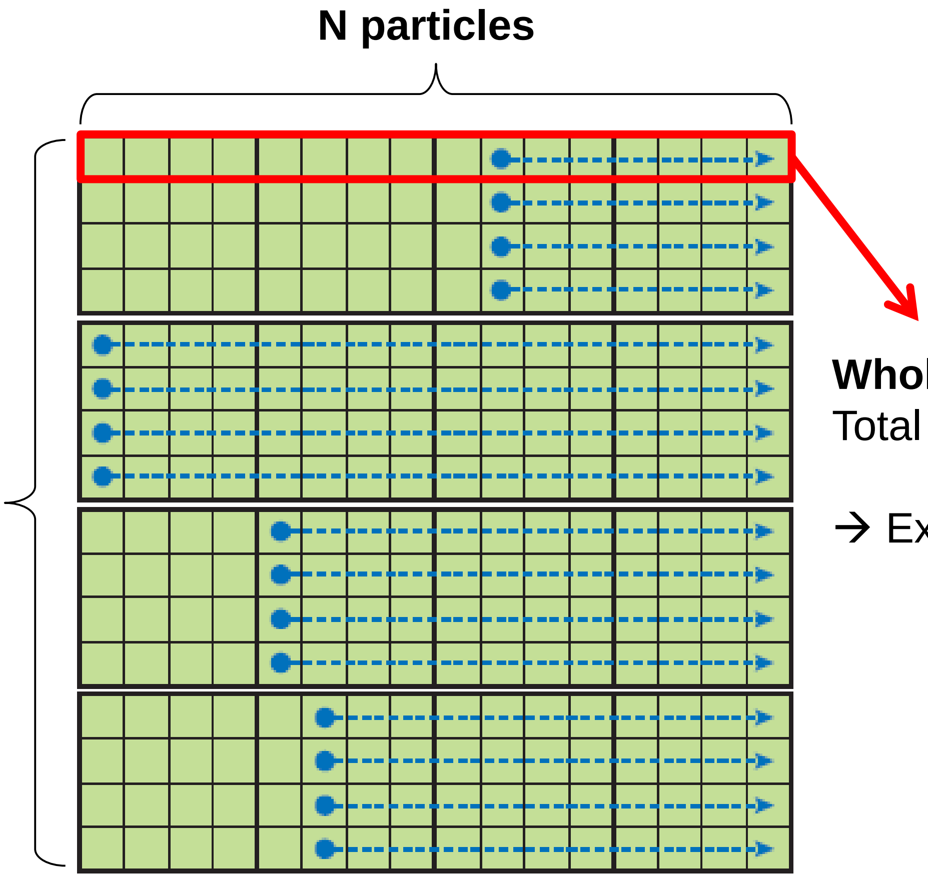
CUDA Implementation

Used algorithm: fast N-Body simulation (Nyland et. Al)



CUDA Implementation

Used algorithm: fast N-Body simulation (Nyland et. Al)

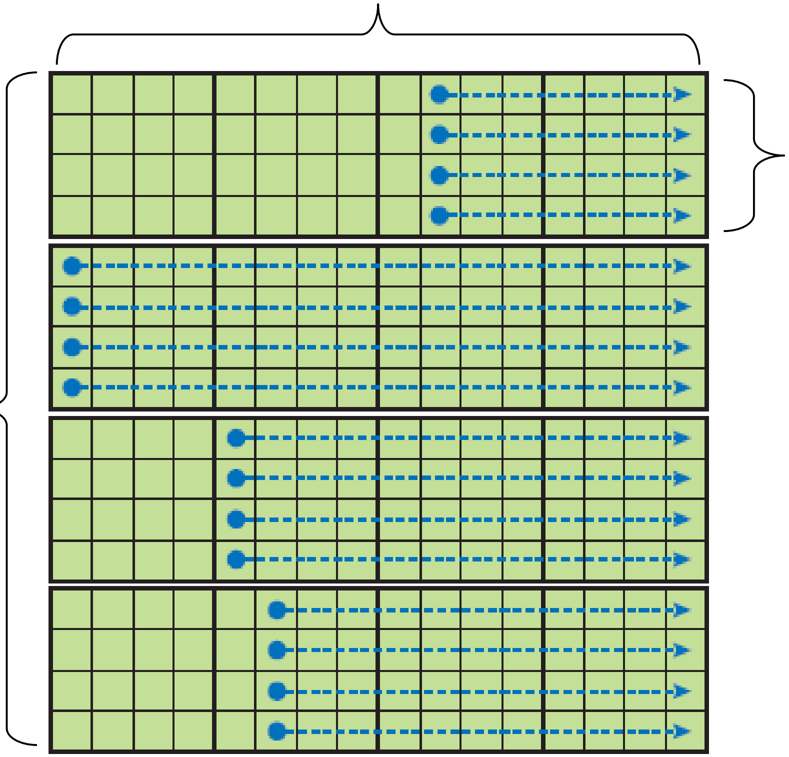


Whole line:
Total force on one particle

→ Executed by one thread

CUDA Implementation

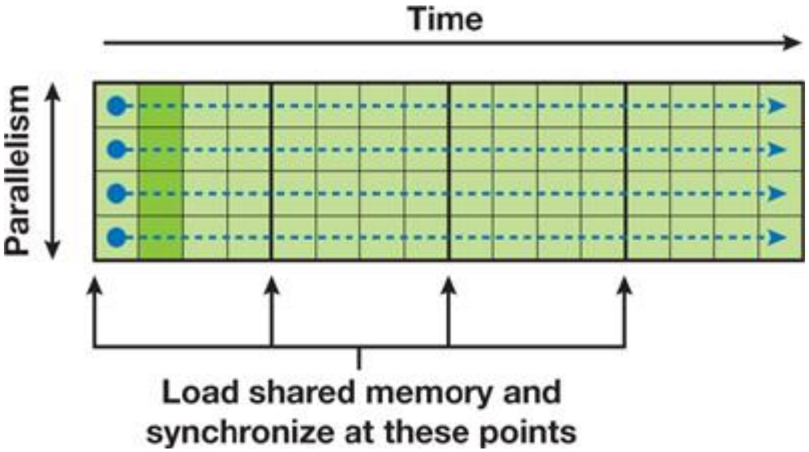
Used algorithm: fast N-Body simulation (Nyland et. Al)



N particles

One block,
blocksize: p threads

N particles,
N/p blocks



CUDA/OpenGL Interoperability

What is OpenGL?

- API to render 2D and 3D graphics scenes.
- Provides interface for drawing basic graphics primitves (e.g. vertex, line, triangle)

Visualization of particles:

- Position of particles in 3D
- Time evolving system

Why do we need interoperability of OpenGL and CUDA?

- Update positions smoothly
- CUDA buffers not visible to OpenGL
- Avoid copying data to CPU

CUDA/OpenGL Interoperability

OpenGL
(Graphics)

Memory Managment

CUDA
(Simulation)

- glBufferData(…)

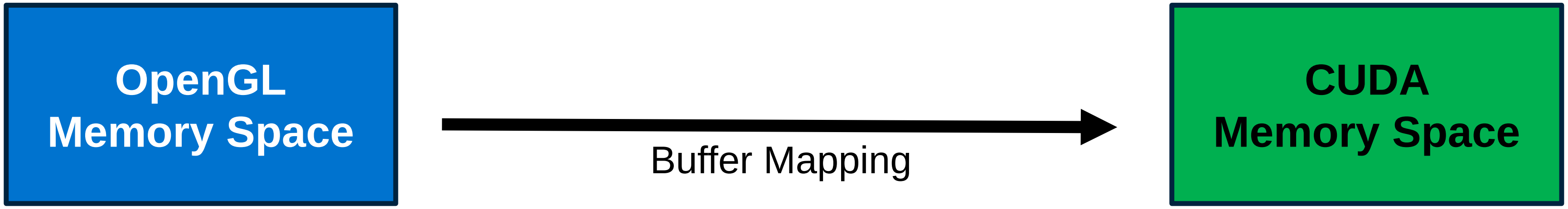
- cudaMalloc(…)

Computations

- Shader (GLSL)

- CUDA Kernel (C/C++)

Interoperabilty



OpenGL Memory Space

Buffer Mapping

CUDA Memory Space

CUDA/OpenGL Interoperability

OpenGL
(Graphics)

CUDA
(Simulation)

Memory Managment

- `glGenBuffer (…)`
- `glBufferData (…)`

Interoperabilty

- `cudaGraphicsGLRegisterBuffer(…)`

→ Buffer Mapping

- `cudaGraphicsMapResources(…)`
- `cudaGraphicsResourceGetMappedPointer(&ptr,…)`

Computations

- `kernel<<<…>>>(ptr, …)`

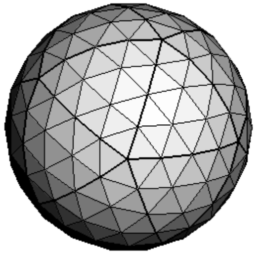
Rendering

- `cudaGraphicsUnmapResources(…)`
- `glDrawArrays(…)`

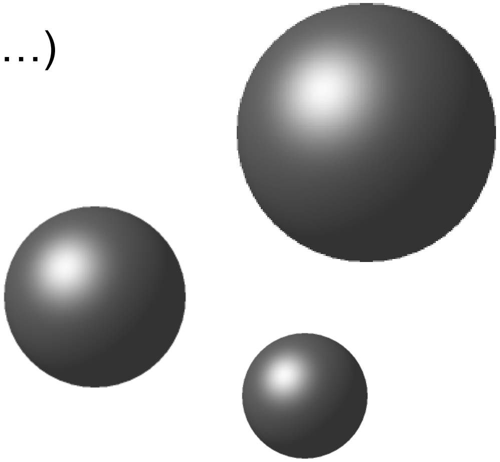
OpenGL

Particle Visualization in OpenGL:

- No default geometric shapes usable with OpenGL interoperability



- Define vertex meshes:
  calculate vertices, normals, faces, scale, move, …
- Define **shader**:
  Program to update each pixel of screen (lighting, color, …)



- Implement extended visibility:
  - enable zooming and
  - rotating of scene,
  - enable window resizing

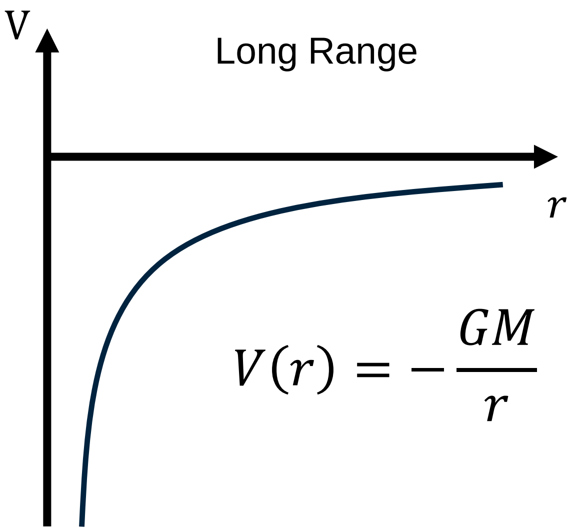
Gravity vs. MD simulation

Similarities:

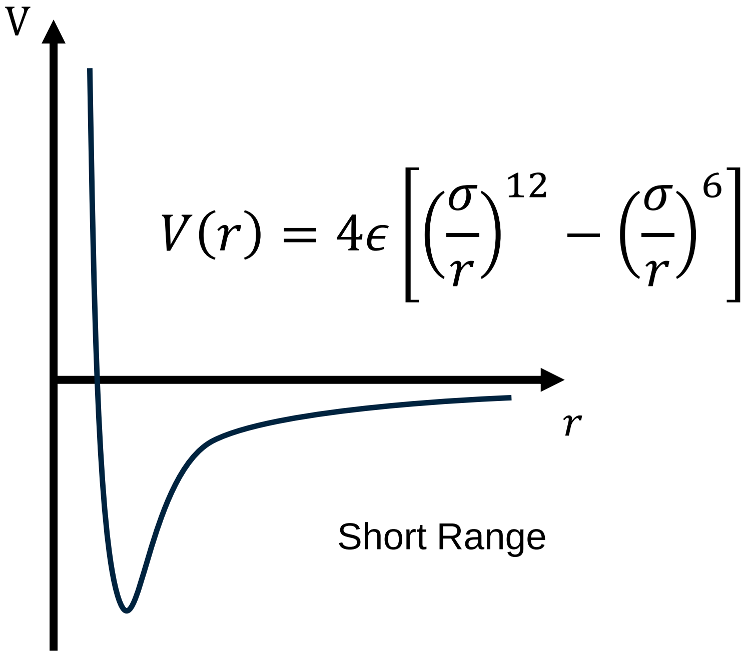
- N-Body systems
- Algorithm (fast N-Body simulation)
- Dynamics according to Newtons law

Differences:

- Interaction Potential:



$$V(r) = -\frac{GM}{r}$$



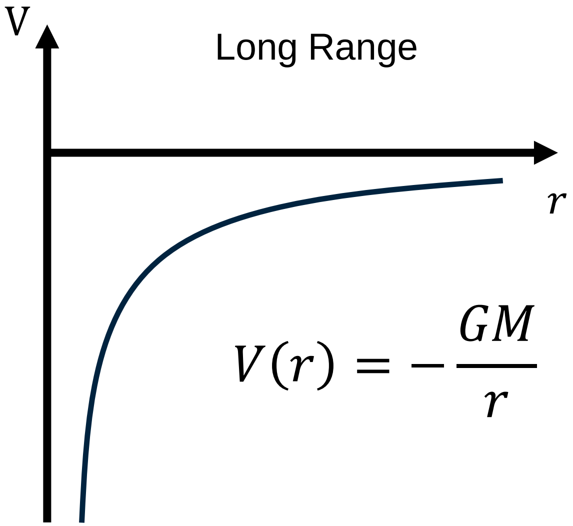
$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

$$\vec{F}(r) = -\vec{\nabla}\, V(r)$$

Gravity vs. MD simulation

Similarities:

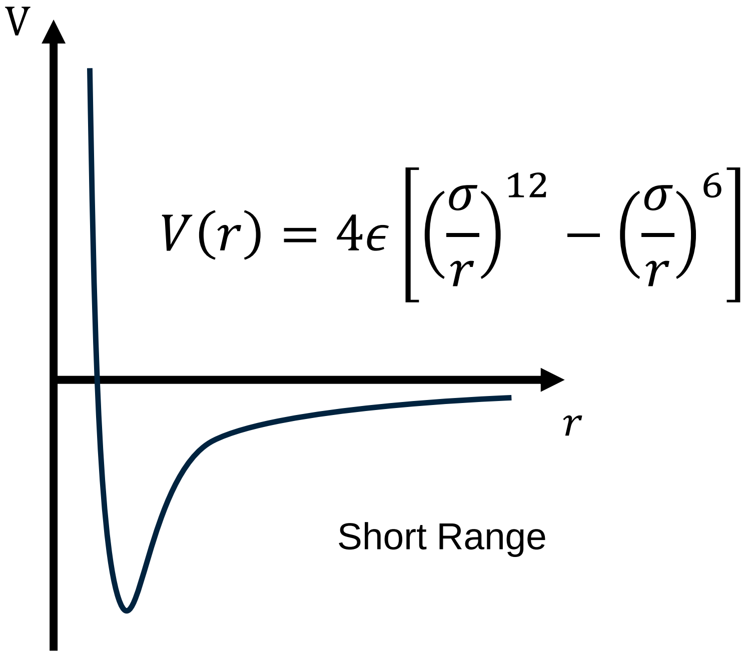
- N-Body systems
- Algorithm (fast N-Body simulation)
- Dynamics according to Newtons law



$$V(r) = -\frac{GM}{r}$$

Differences:

- Interaction Potential:



$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

$$\vec{F}(r) = -\vec{\nabla}\, V(r)$$

Test Scenarios

- Enjoy!