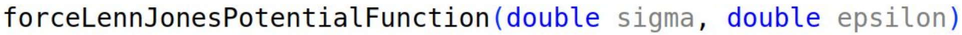
Collision of two bodies

Main Tasks:

- include googletest, spdlog
- add CI pipeline
- new way of calculating forces
- performance measurement
- reading files with cuboids and creating the particles accordingly
- the actual simulation

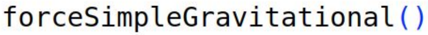
Lennard-Jones-Potential using some :-)

- calculateF() takes function parameter for force calculations between two Particles
- possible to dynamically switch between the two force calculations
- “curried” Function for Lennard-Jones-Potential



```
forceLennJonesPotentialFunction(double sigma, double epsilon)
```

$$-\frac{24\cdot\epsilon}{(||x_i-x_j||_2)^2}\left(\left(\frac{\sigma}{||x_i-x_j||_2}\right)^6-2\left(\frac{\sigma}{||x_i-x_j||_2}\right)^{12}\right)(x_i-x_j)$$



```
forceSimpleGravitational()
```

$$\frac{m_i\cdot m_j}{(\|x_i-x_j\|)^3}(x_j-x_i)$$

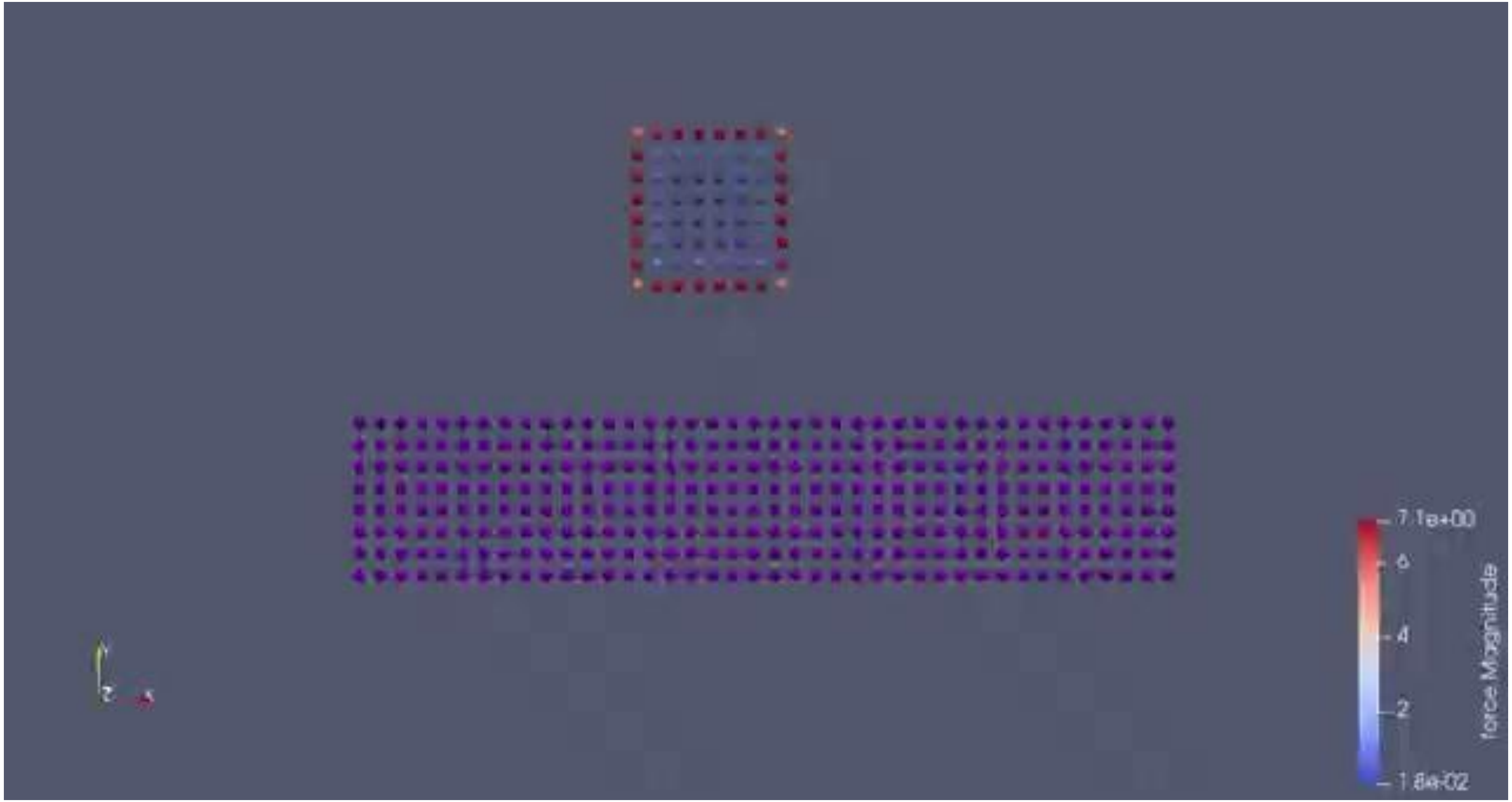
Performance

Parameters:

- `std::chrono::high_resolution_clock` used for time measurement
- Measurements in Linux environment on AMD Ryzen 7 5700U
- Compiled with gcc and flag -O2 for runtime measurement
- Memory access measured with cachegrind (program compiled with -O1)

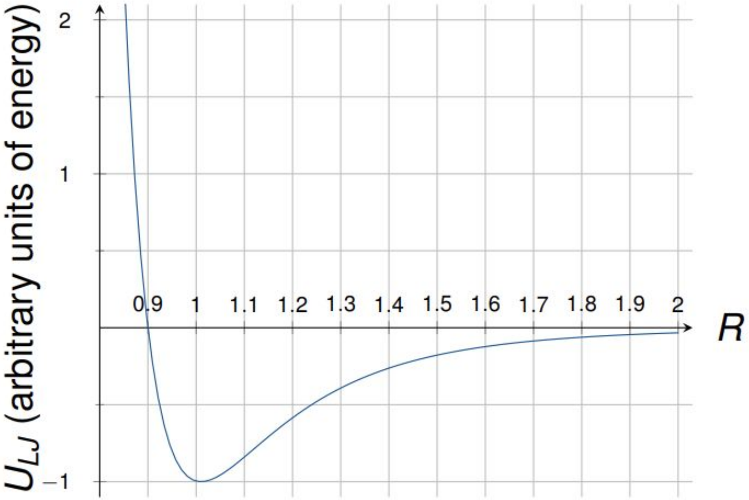
	nested loop	Newtons third law
runtime	~ 122 sec	~ 114 sec
memory access	~ 5,86 bil. data accesses	~ 3,27 bil. data accesses
cache miss rate	0.1 %	0.1 %

Simulation

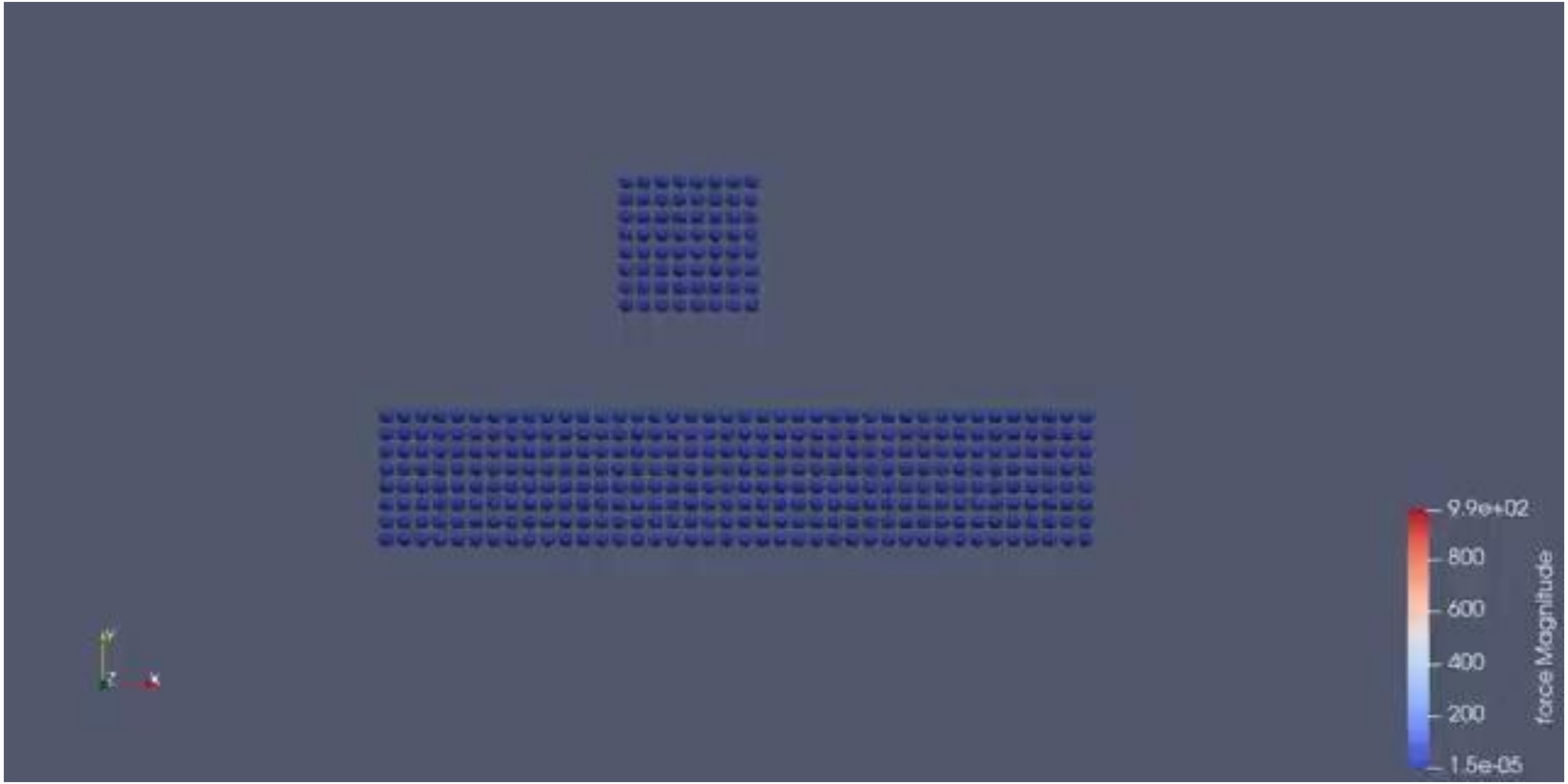


Observations

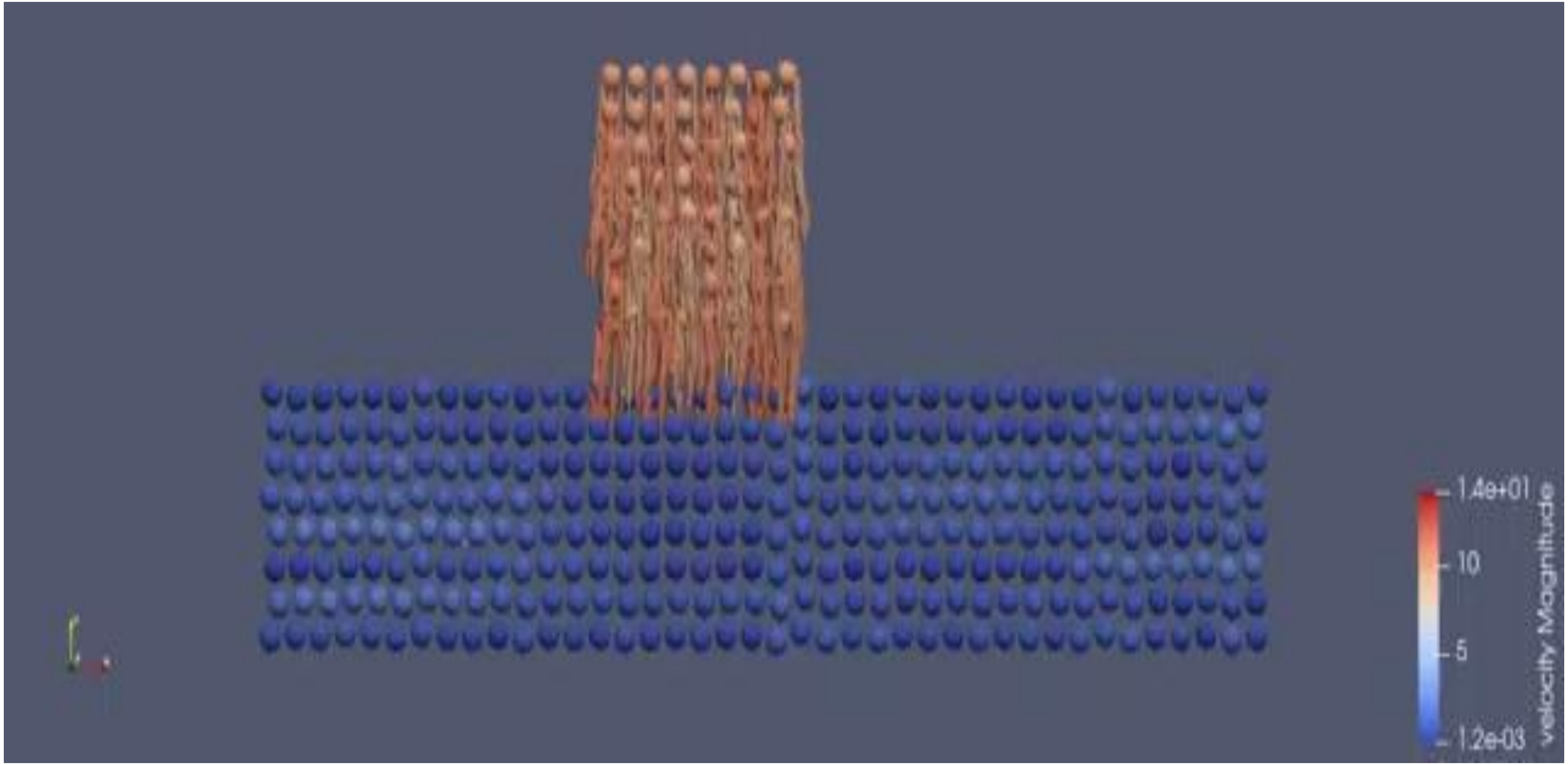
- particles push each other during collision
- force of isolated particles oscillates



Simulation



Simulation

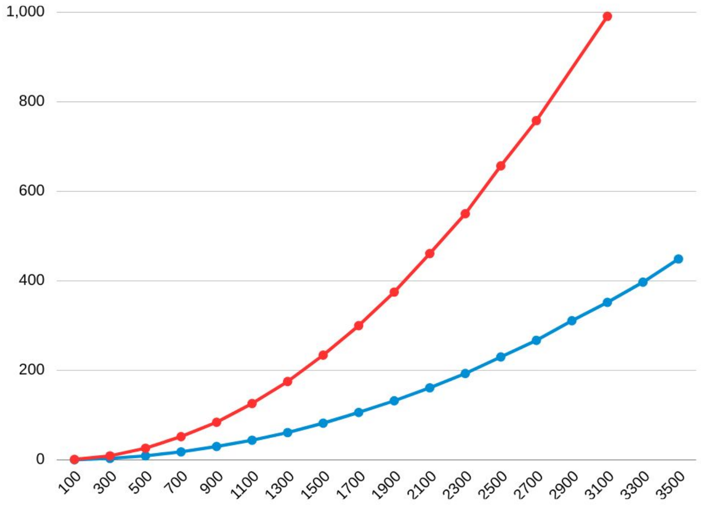


Calculating force

- utilising Newton’s third law $F_{ij} = -F_{ji}$

- only half of the particle combinations needed

P	0	1	2	3	4	5
0	**x**	✔	✔	✔	✔	✔
1	✔	x	…			
2	✔		x			
3	✔			x		
4	✔				x	
5	✔					x



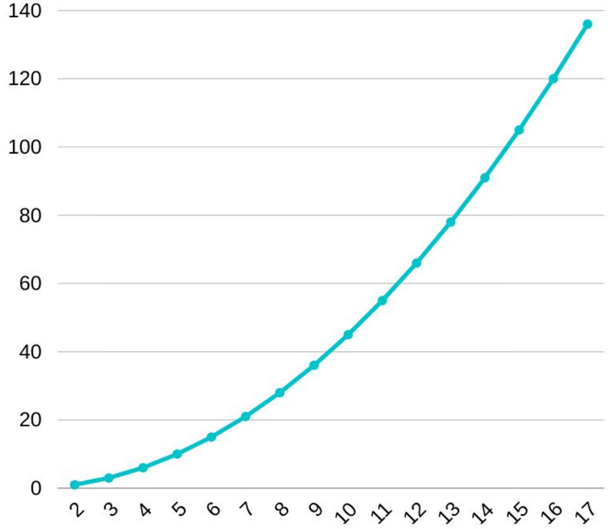
Calculating force

- amount of force calculations non-linear

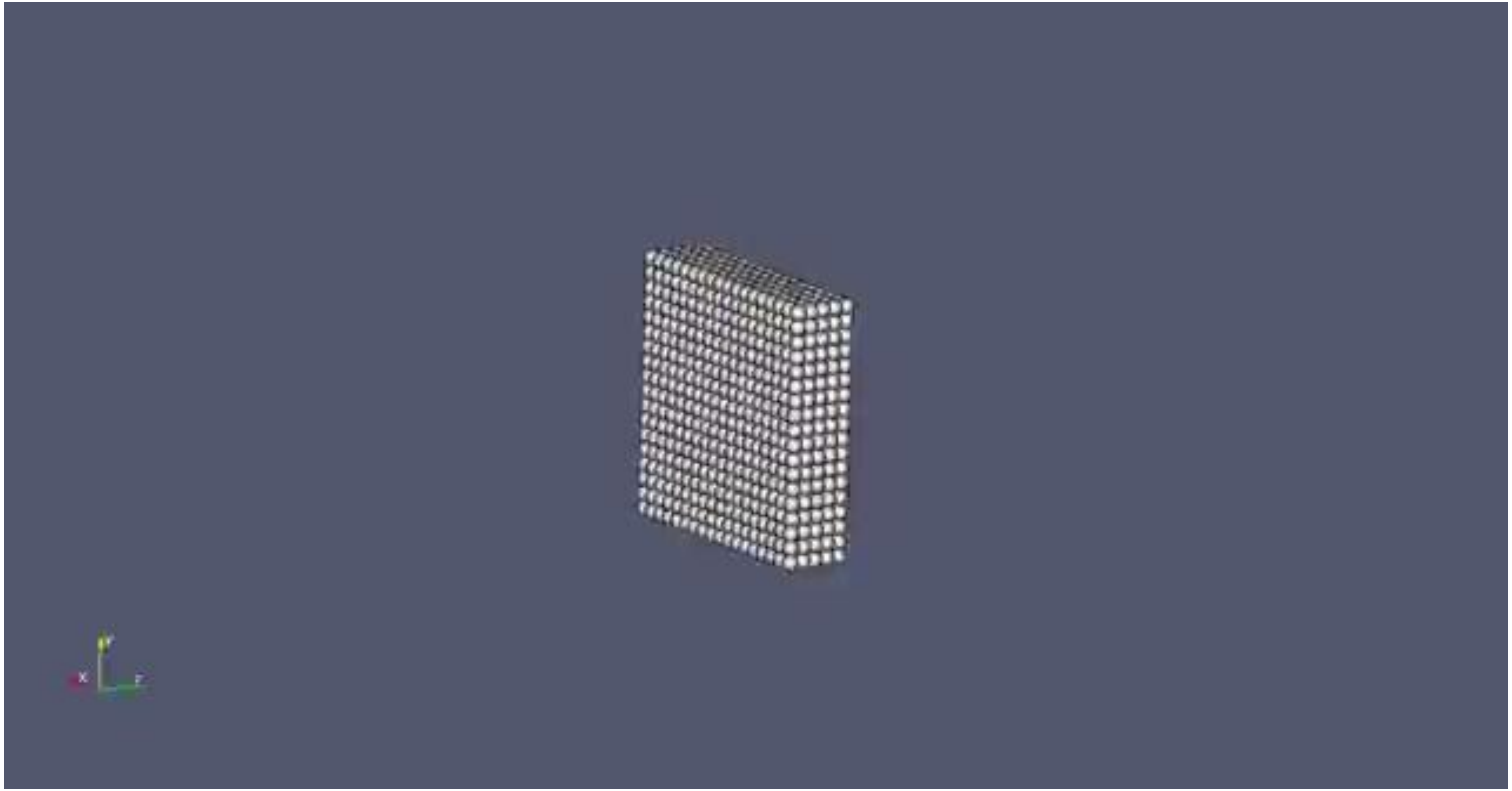
  cuboid: 20 × 40 × 4 = 3200 ➪ 5.1 mio

  cuboid: 20 × 35 × 4 = 2800 ➪ 3.9 mio

  ➪ handshake problem



Smashing stuff



Smashing stuff