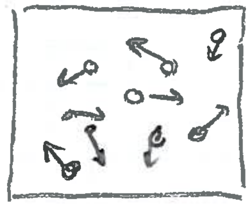
Let's look at a simple many-particle system

<u>Hard spheres in a (periodic) box</u>



billiards!

- parameters N, d_1, d_2 $(\rho = \frac{N}{d_1 d_2})$, m, E, R
- variables $r_1, r_2, \ldots r_N$, $v_1, v_2, \ldots v_N$ } $O(N)$ variables

N is large (but can't be $\sim N_A$)

- initial conditions? equilibration?
- run: <u>do time evolution</u>
  - instantaneous collisions like in Lorentz gas
  - calculate and compare collision times

Problem: there are $N \cdot (N-1)$ collision times
& $4N$ border crossing times
Want to run for TN, $T \gg 1$ collisions

(1) naive approach: calculate it all, then update

memory usage (store variables) $O(N)$
cpu calculate coll. times $O(N^2)$
× number of time steps $\cdot N \cdot T$ $= O(N^3)$

(2) better approach: store event times

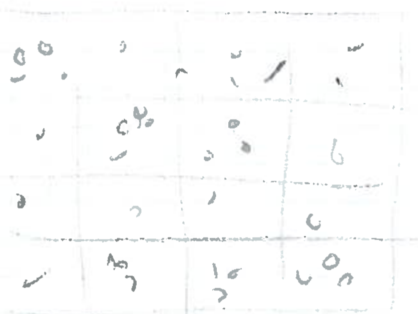
- recalculate only the ones that change

memory usage $O(N^2)$ (all event times)

cpu: recalculate at every step for only 2 particles
$O(N) \cdot NT = O(N^2)$ = much cheaper

(3) Further optimisation

divide into cells, so even fewer new calculations



SCALING OF CPU/MEMORY MATTERS
(because it's Stat Phys & N_A is so big)

Use cachegrind to see where clock cycles are going.

MD problems we will deal with

1. real interactions are more complicated.

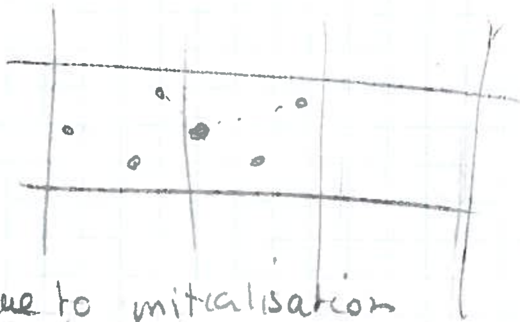
Quantum Mechanics; electron clouds of atoms interact
too much work if you want to do many particles

⇒ + realistic effective classical force fields / potentials.
- combination: Quantum MD (still pretty expensive)

⇒ unlike in previous examples usually smooth interaction

2. $N \leq N_A$, $t \leq 1s$ $d << 1m$

periodic boundary conditions
(nearest image convention)
be careful if correlation length > d



finite-size effects

averaging over finite time ⇒ errors due to initialisation
& statistical errors

3. Integration algorithms have finite accuracy

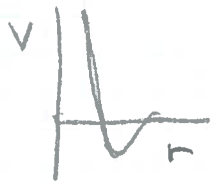
Example: smooth interactions in a gas

hard spheres & instantaneous coll ⟷ low density.
(no 3-particle interactions)

more realistic
soft repulsion

3 particles close together interact

attractive interaction
(f.e. van der Waals)



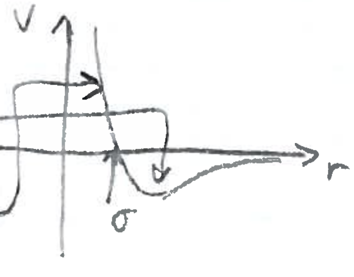
Your first simulation project: Argon liquid
dense, and can actually solidify

Lennard-Jones $V(\vec{r}_1 - \vec{r}_2) = U_{LJ}(|\vec{r}_1 - \vec{r}_2|)$

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

weak attractive part

strong repulsive part



$U_{LJ} = 0$ at σ

Minimum at

$$\frac{\partial U_{LJ}}{\partial \vec{r}} = 0 = 4\epsilon \left[12\left(\frac{\sigma}{r}\right)^{11} - 6\left(\frac{\sigma}{r}\right)^{5} \right] \frac{\partial}{\partial \vec{r}}\left(\frac{\sigma}{r}\right)$$

$$\Rightarrow 2\left(\frac{\sigma}{r}\right)^6 = 1 \quad \Rightarrow r_{min} = \sigma\, 2^{1/6}$$

$$U_{LJ}(r_{min}) = -\epsilon$$

LJ very common ; good for argon too

$\epsilon = 1{,}654 \cdot 10^{-21}$ J $\quad \sigma = 3{,}405$ Å

+ minimum image convention + integration = your simulation

Note on efficiency when you calculate forces

$$F_{LJ} = -\frac{\partial U}{\partial r} = 4\epsilon \left[12 \frac{\sigma^{11}}{r^{11}} - 6\frac{\sigma^5}{r^5} \right] \sigma \frac{\vec{r}}{r^3}$$

$$= 4\epsilon \left[12 \frac{\sigma^{12}}{r^{14}} - 6 \frac{\sigma^6}{r^8} \right] \vec{r}$$

even powers, no expensive $\mathrm{sqrt}((r_i - r_j)^2)$

Argon liquid model: LJ, N atoms, periodic boundaries

Variables: $(r_1, r_2, \dots r_N, v_1, v_2, \dots v_N) = \gamma$

Eqs of motion:

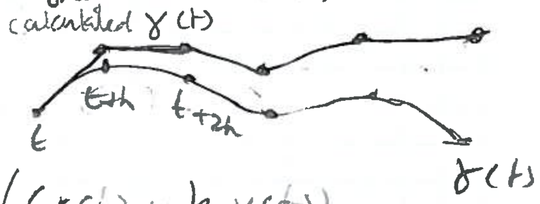
$$\boxed{\begin{aligned} \dot{r}_i &= v_i \\ \dot{v}_i &= -\sum_j \frac{\partial V(\vec{r})}{m_i \partial \vec{r}}\Big|_{\vec{r}=\vec{r}_i-\vec{r}_j} = a_i \end{aligned}} \qquad \dot{\gamma} = f(\gamma, t)$$

How to solve? Small time steps h

Naive approach: $\gamma(t+h) = \gamma(t) + h\, f(\gamma, t) + \mathcal{O}(h^2)$

$$r(t+h) = r(t) + h\, v(t)$$

$$v(t+h) = v(t) + h\, a(r(t))$$



$$E(t+h) = \tfrac{1}{2}m[v(t) + h\, a(r(t))]^2 + V(r(t) + h\, v(t))$$

$$= \tfrac{1}{2}m\, v(t)^2 + \cancel{m\, h\, a(r(t))} + \tfrac{1}{2}m\, h^2 a(r(t))^2$$

$$+ V(r(t)) + \cancel{h\, v(t)\, V'(r(t))} + \tfrac{1}{2}(h v(t))^2\, V''(r(t))$$

$$= E(t) + \mathcal{O}(h^2) \Rightarrow \text{after long time error } \mathcal{O}\left(\frac{t}{h} \cdot h^2\right) = \mathcal{O}(ht)$$

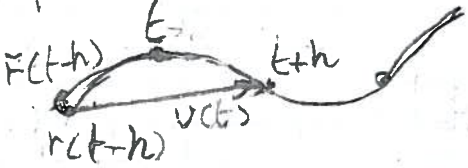
This is terrible, very unstable <u>DO NOT USE</u>

Do better (Verlet algorithm)

$$r(t+h) = r(t) + h\, v(t) + \tfrac{1}{2}h^2 a(t) + \tfrac{1}{6}h^3 \dot{a}(t) + \mathcal{O}(h^4)$$

$$v(t+h) = ?? =$$

one step forward and one back



$$r(t-h) = r(t) - h\, v(t) + \tfrac{1}{2}h^2 a(t) - \tfrac{1}{6}h^3 \dot{a}(t) + \mathcal{O}(h^4)$$

backwards $\neq$ inverse(forwards)

$+$

$$\boxed{\begin{aligned} r(t+h) &= -r(t-h) + 2\, r(t) + h^2 a(t) + \mathcal{O}(h^4) \quad &(1) \\ v(t) &= \frac{r(t+h) - r(t-h)}{2h} \quad &(2) \end{aligned}}$$

Over long time, divergence? by induction $err(1) = \mathcal{O}(h^4)$ in position ($\mathcal{O}(h^2)$ in velocity)

$$err(n+1) = err(n-1) + 2\, err(n) + h^2\, err(n) + \mathcal{O}(h^4) \qquad n = \frac{t}{h}$$

$$\frac{(n+1)(n+2)}{2} = -\frac{(n-1)\,n}{2} + \frac{2n(n+1)}{2}$$

$$\Rightarrow \text{error after time } t \approx \left(\frac{t}{h}\right)^2 \mathcal{O}(h^4) = t^2 h^2$$

<u>end lecture 2</u>