N-body simulation

Part 1: introduction

epcc

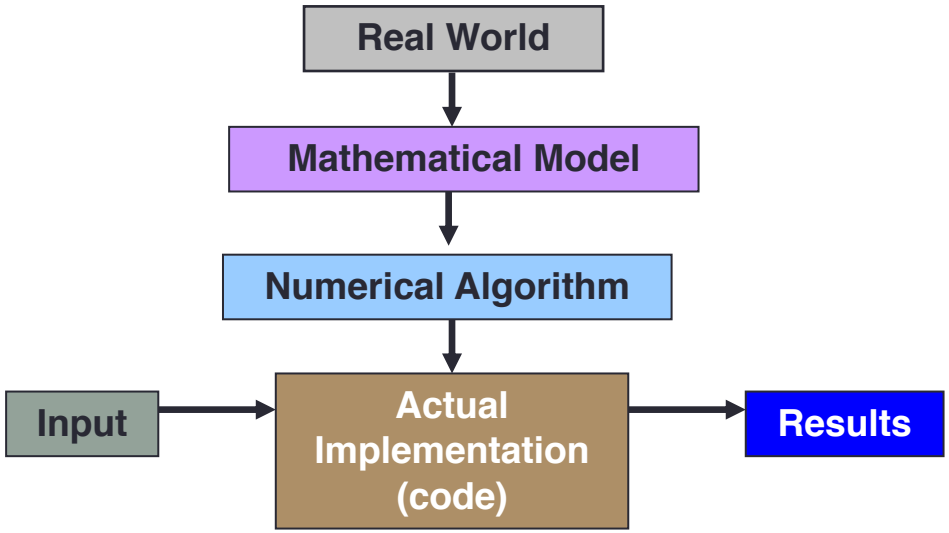
THE UNIVERSITY OF EDINBURGH

Overview

- What is N-body simulation and what is it used for?
- Underlying physics & mathematical formulation
- General N-body simulation algorithm
- Computational cost and big O notation

What is N-body simulation?

- Particles interact through physical forces
- A mathematical model describe these interactions and how particles move
- A numerical algorithm derived from the mathematical model allows us to simulate the motion of N interacting particles - a dynamical system

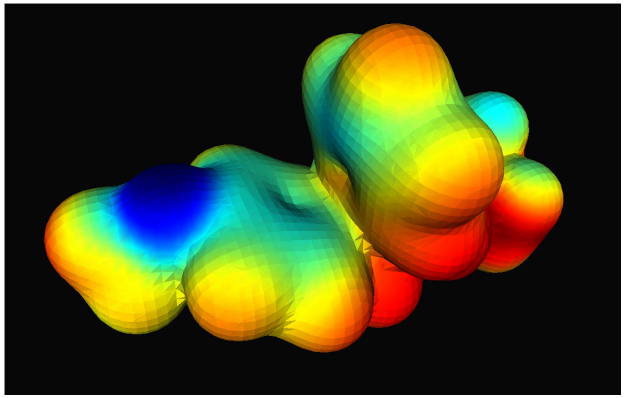


epcc

THE UNIVERSITY OF EDINBURGH

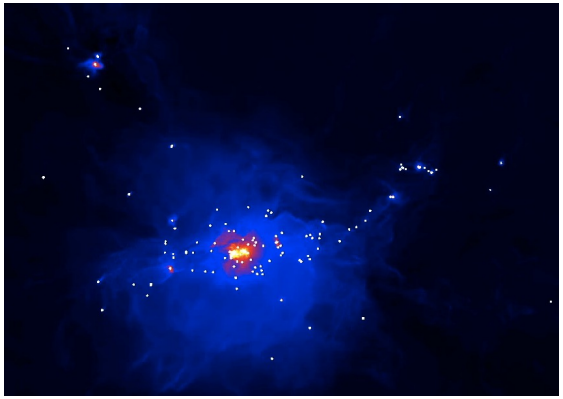
Science that uses N-body simulation

- Atomistic simulation / molecular dynamics:
  - Biomolecular modelling & simulation
  - Computational chemistry
  - Materials science



credit RPI/Curt Breneman

- Astrophysical N-body simulation:
  - Orbital dynamics
  - Formation & evolution of planetary systems
  - Cosmology

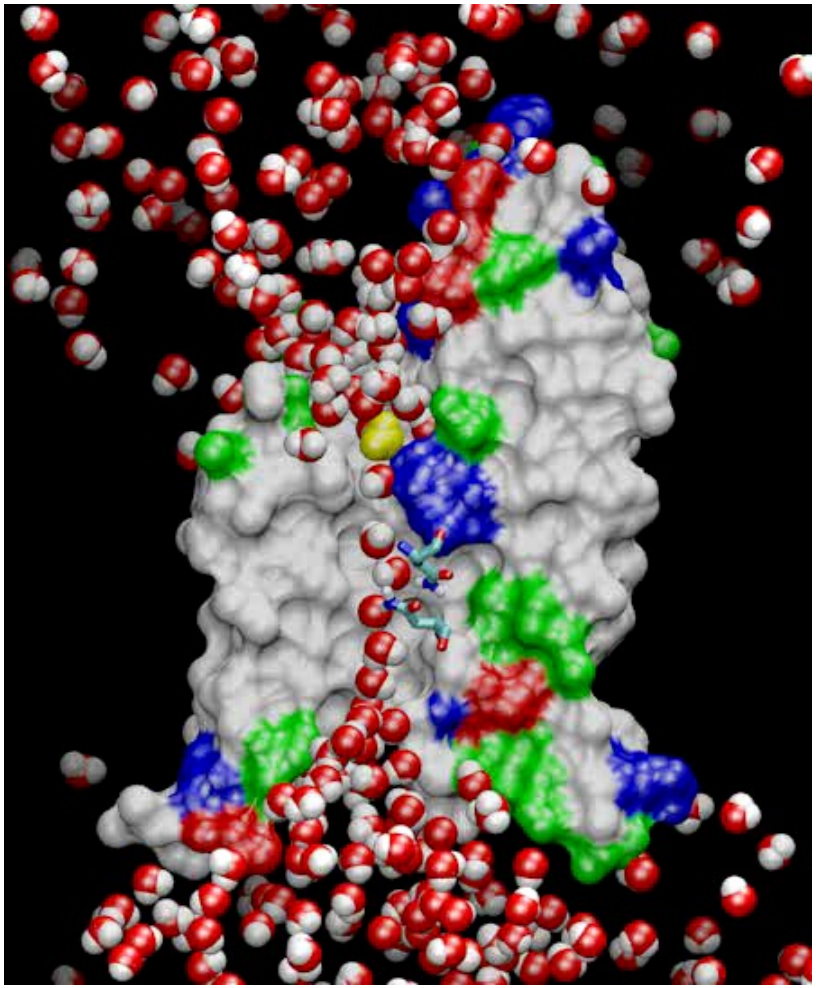


Matthew Bate, U. Exeter

epcc

THE UNIVERSITY OF EDINBURGH

Water permeation in an aquaporin



Tajkhorshid, E., et al. Science **296**:525-530 (2002)

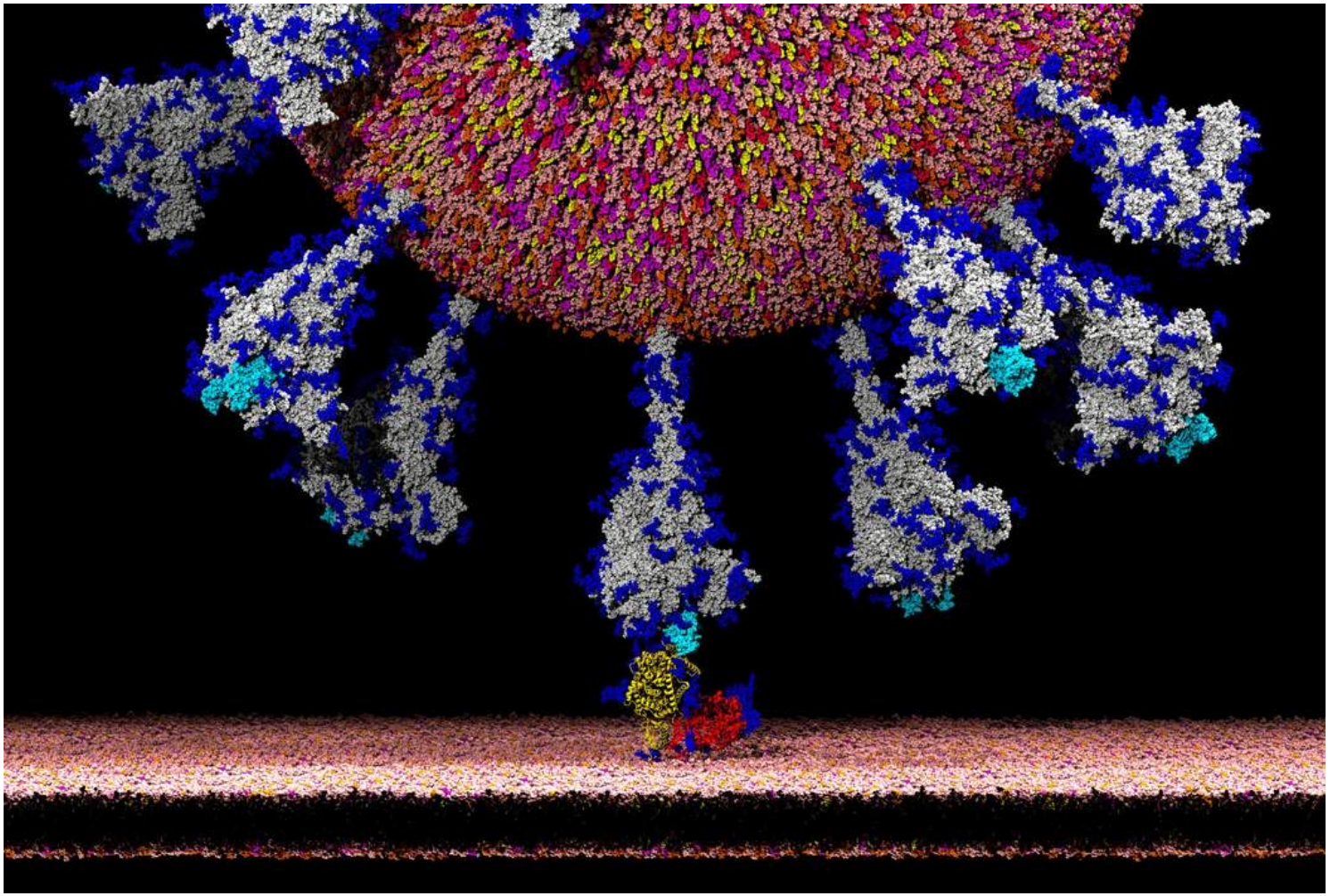
Star formation: Density

epcc

Matthew Bate – University of Exeter

THE UNIVERSITY OF EDINBURGH

SARS-CoV-2



Credit: Lorenzo Casalino, Amaro Lab, U.C. San Diego

What is a particle?

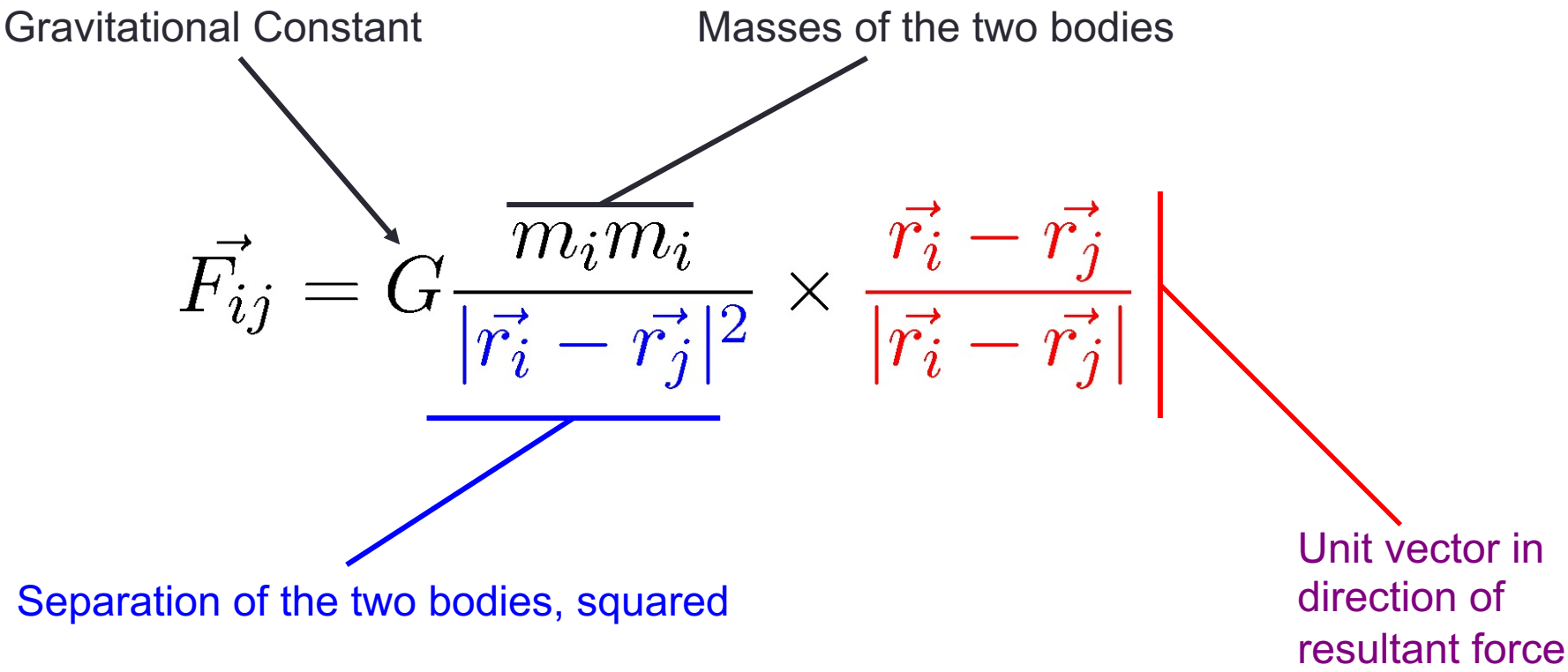
- Asteroid, planet, star (astrophysical n-body simulation)
  - To model solar systems, globular star clusters, colliding galaxies, …
- Atom (atomistic simulation, molecular dynamics)
  - To model liquids, solids (e.g. crystals), gases, molecules, viruses
- Lots of things inbetween (coarse-grained / multi-scale simulation)
  - molecule
  - protein
  - small volume of gas, liquid, or solid (10^X atoms)

Interactions

- Gravity (astrophysical simulation)
- Electrostatic (atomistic simulation)
- Use simple approximate descriptions of forces
  - *Accurate enough* for simulation purpose
  - *Fast enough* to compute in useful time
- Include approximate quantum effects where essential
  - 'Traditional' N-body simulation uses Newton's Laws – classical physics
  - Describing motion of electrons requires full quantum physics

Example: gravitational force

- Newton's Universal Law of Gravitation
  - Force between two bodies labelled *i* and *j* (in vector form)



epcc

THE UNIVERSITY OF EDINBURGH

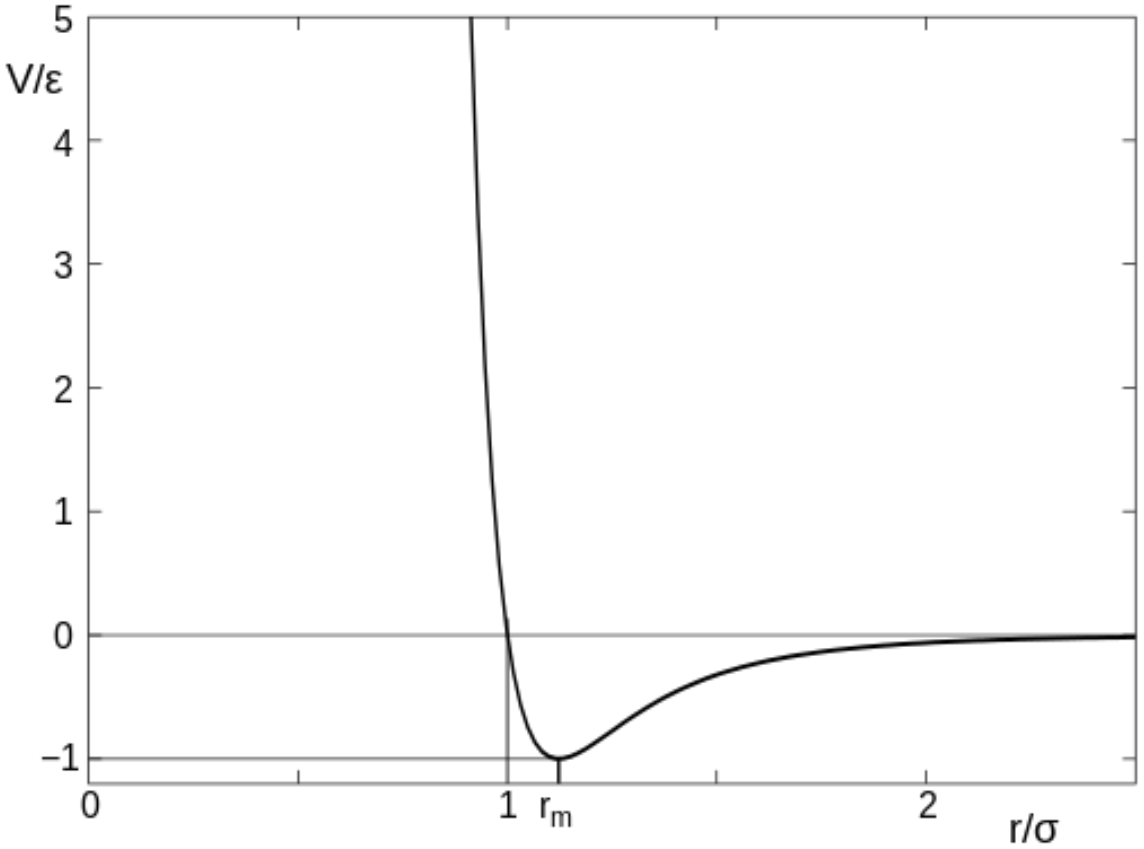
Example: Lennard-Jones potential

- Electrostatic attraction ($1/r^2$ like gravity) dominates between atoms further apart than r_m
- Repulsion dominates between atoms closer together than r_m

$$V(r_{ij}) = 4\varepsilon_{ij}\left(\left[\frac{\sigma_{ij}}{r_{ij}}\right]^{12} - \left[\frac{\sigma_{ij}}{r_{ij}}\right]^{6}\right)$$

$$r_m = 2^{\frac{1}{6}}\sigma$$

$$F = -\frac{dV}{dr}$$



epcc

THE UNIVERSITY OF EDINBURGH

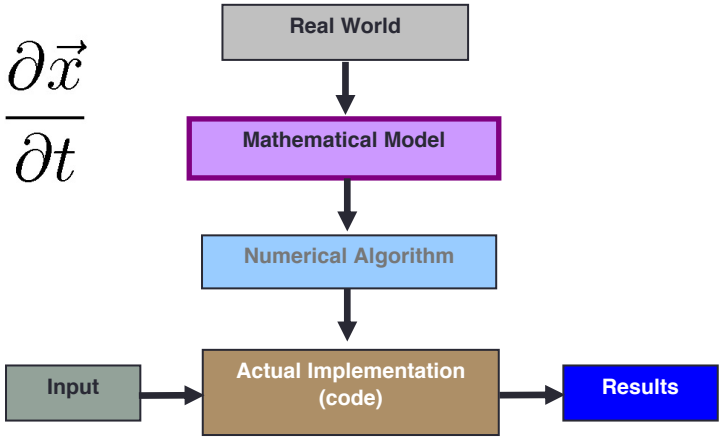
Why N-body simulations?

Classical mechanics

Mathematical model underlying N-body simulation: differential equations expressing Newton's Laws of Motion (i.e. classical mechanics):

$$\vec{F} = m\vec{a} \qquad \vec{a} = \frac{\partial \vec{v}}{\partial t} = \frac{\partial^2 \vec{x}}{\partial t^2} \qquad \vec{v} = \frac{\partial \vec{x}}{\partial t}$$

These apply for each of the N particles



If we define forces and initial conditions (positions & velocities), can **in principle** solve these equations

epcc

THE UNIVERSITY OF EDINBURGH

Real problems are difficult to solve

- Simple problem: apple falling (1 body)
  - Can solve exactly analytically
  - Actually a simplified 2-body problem (apple & Earth)
- Less simple: planet orbiting a star (2 bodies)
  - Can still be solved exactly
- Three body problem
  - No analytic solution *in general*
- Most real problems too complex for exact analytical solution
  - Applies to general n-body problems, where n ≥ 3

➔ **need to solve numerically using a computer**

epcc

THE UNIVERSITY OF EDINBURGH

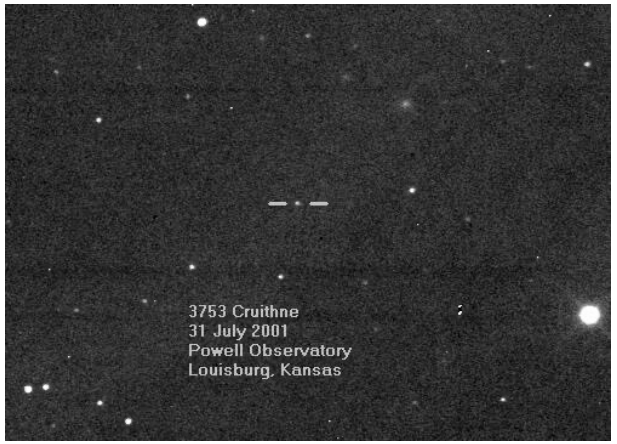
Earth's "second moon"

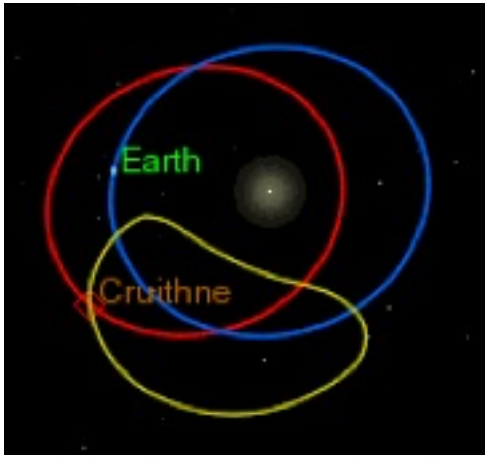
"3753 Cruithne"

- First observed 10 Oct 1986
- Near-Earth asteroid, 5km diameter

Orbit not determined until 1997

- Co-orbits the Sun with Earth





Complicated 'kidney bean' orbit:

- Not possible to predict by solving equations of motion exactly
- Required simulation that includes other planets: many-body problem

epcc

THE UNIVERSITY OF EDINBURGH

Solving differential equations numerically

- Mathematical model:

$$\vec{F} = m\vec{a} \qquad \vec{a} = \frac{\partial \vec{v}}{\partial t} = \frac{\partial^2 \vec{x}}{\partial t^2} \qquad \vec{v} = \frac{\partial \vec{x}}{\partial t}$$

- Differential equations are continuous, computers are digital

- ***Discretisation*** converts differential equations into forms suitable for numerical solution (details next lecture)

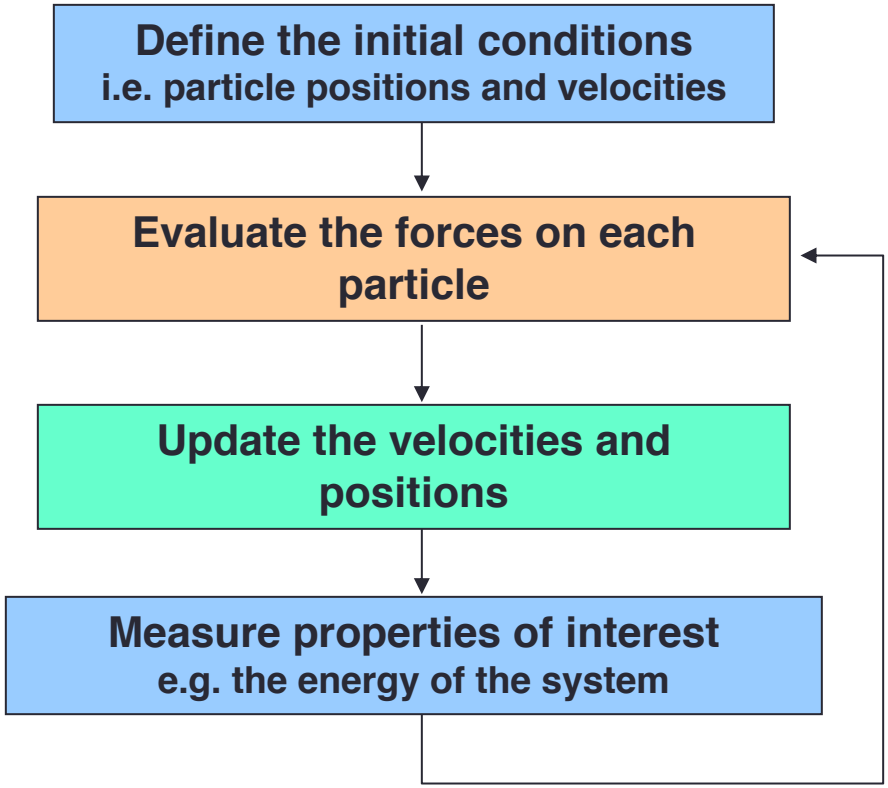
- Solution computed by **time integration**:
  - estimate F $\Rightarrow$ a $\Rightarrow$ v $\Rightarrow$ x $\Rightarrow$ F over small discrete time increments
  - repeat to simulate motion of particles over longer time

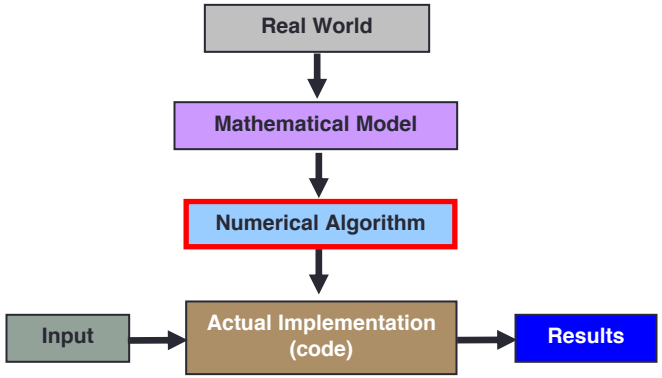
epcc

THE UNIVERSITY OF EDINBURGH

General N-body algorithm

General N-body simulation algorithm



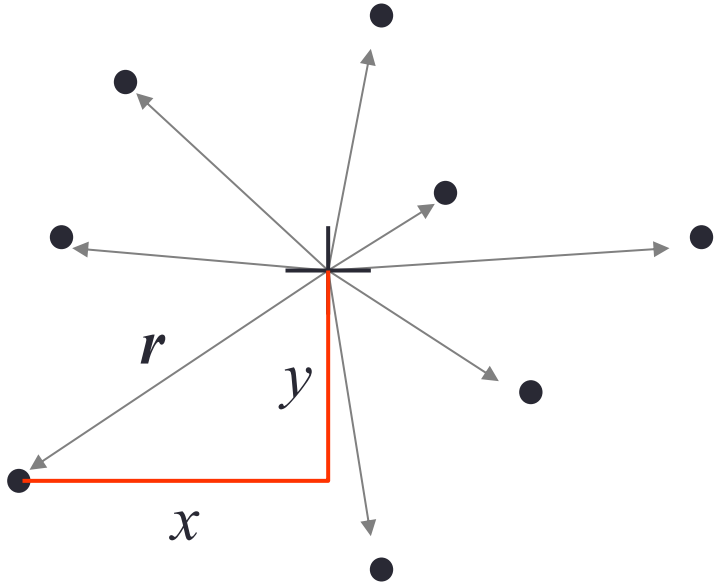


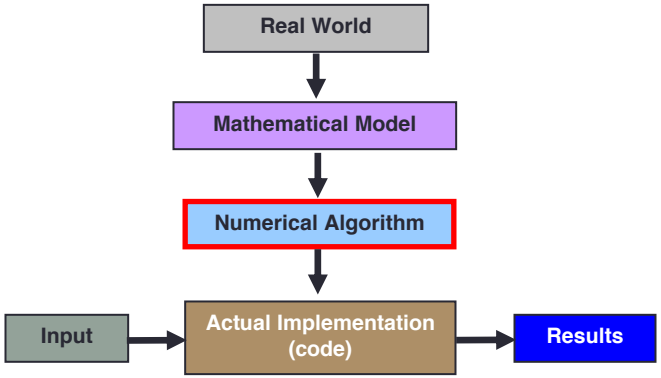
epcc

THE UNIVERSITY OF EDINBURGH

Particle representation: positions

- Positions stored as arrays of real numbers
  - e.g. in 2-D, vector position ***r*** of i = 1 to N particles stored as x[i] and y[i] displacements:



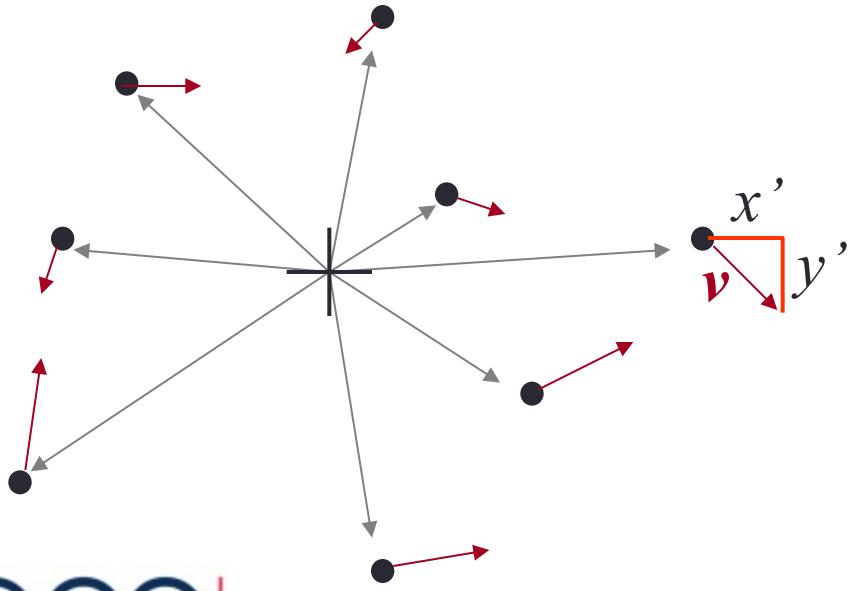


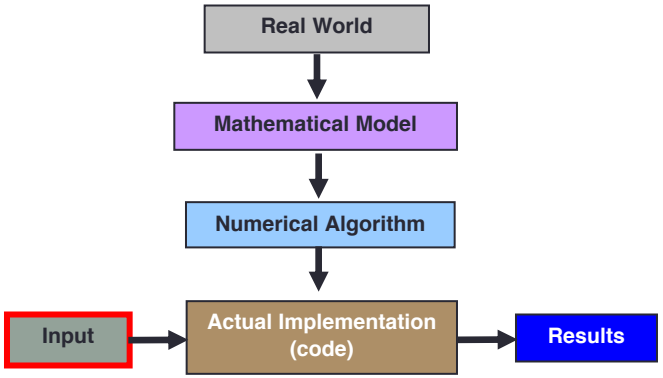
epcc

THE UNIVERSITY OF EDINBURGH

Initial Conditions

- We need initial positions and velocities
  - position ***r*** stored as x and y displacements,
    - e.g. miles, metres.
  - velocity ***v*** stored as x' and y' "speeds":
    - units of displacement per unit time, e.g. mph, m/s.



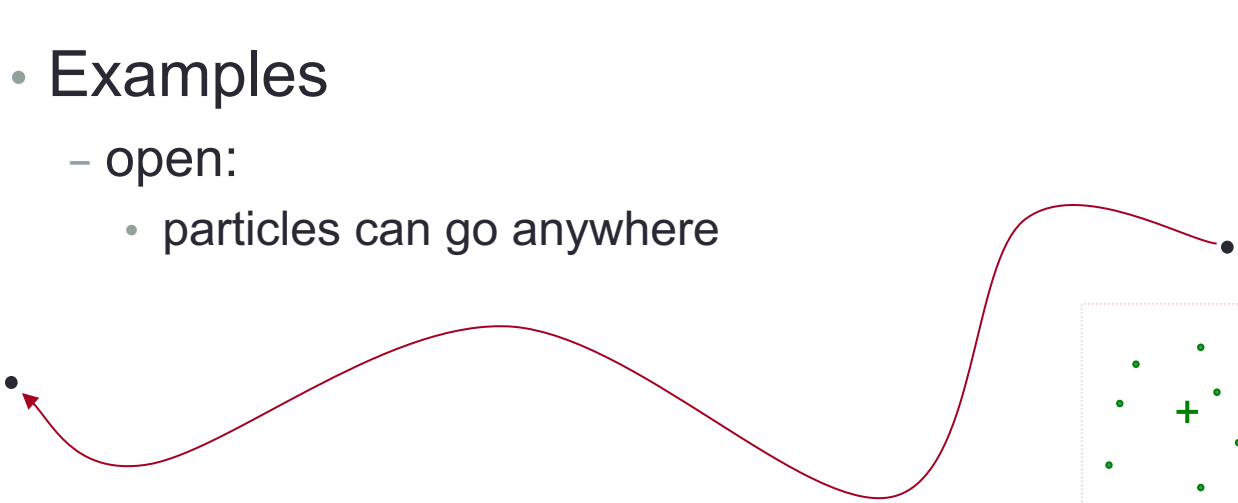


epcc

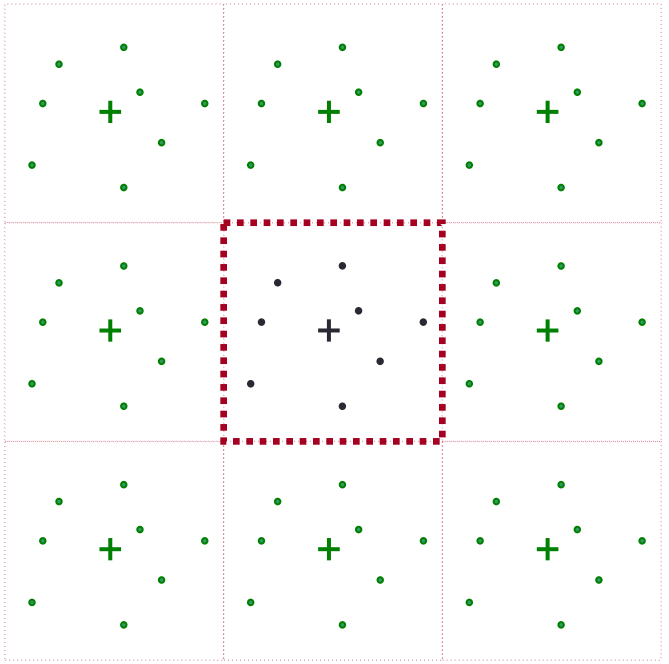
THE UNIVERSITY OF EDINBURGH

The Simulation Domain: Boundaries

- Examples
  - open:
    - particles can go anywhere



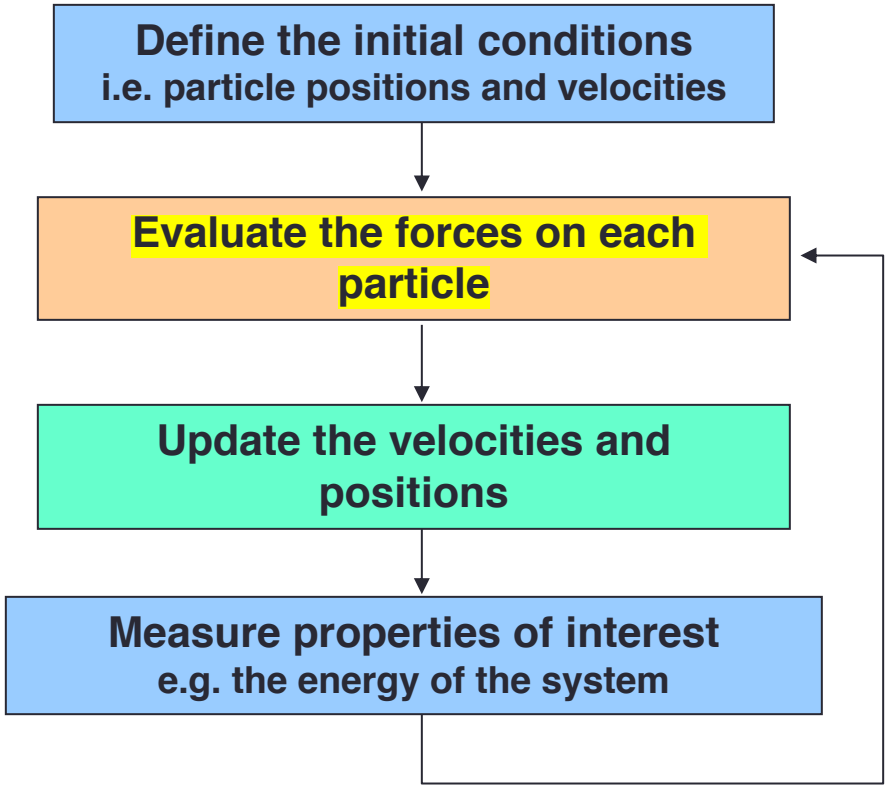
  - periodic:
    - particles stay in a fixed volume, by wrapping around the edges.
    - The system ‘sees’ images of itself tiled out in all directions.
    - suitable for simulating “bulk” behaviour inside of many systems (e.g. gas, biomolecule in water)

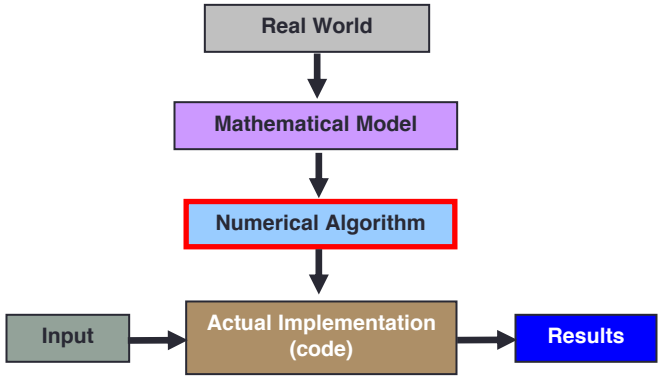


epcc

THE UNIVERSITY OF EDINBURGH

General N-body simulation algorithm



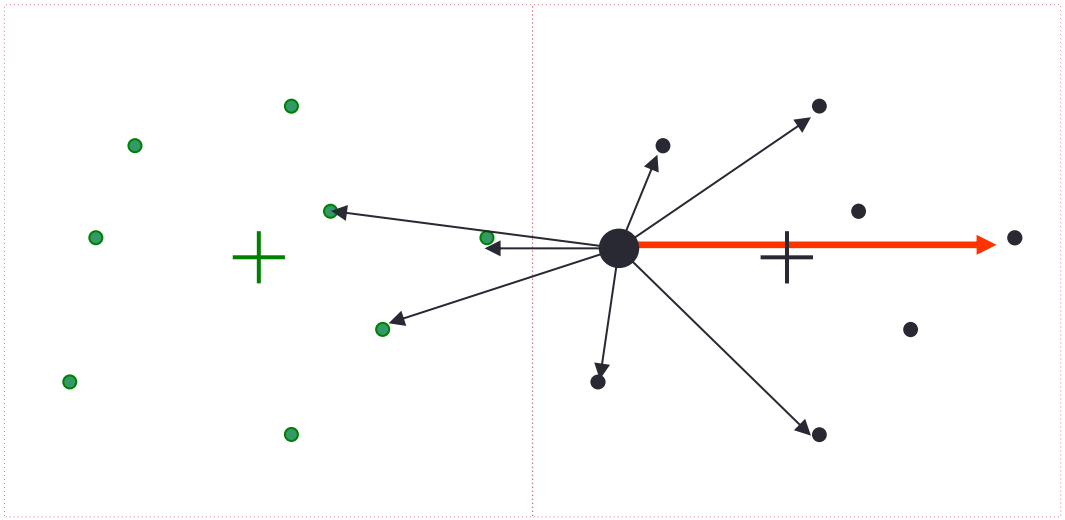


epcc

THE UNIVERSITY OF EDINBURGH

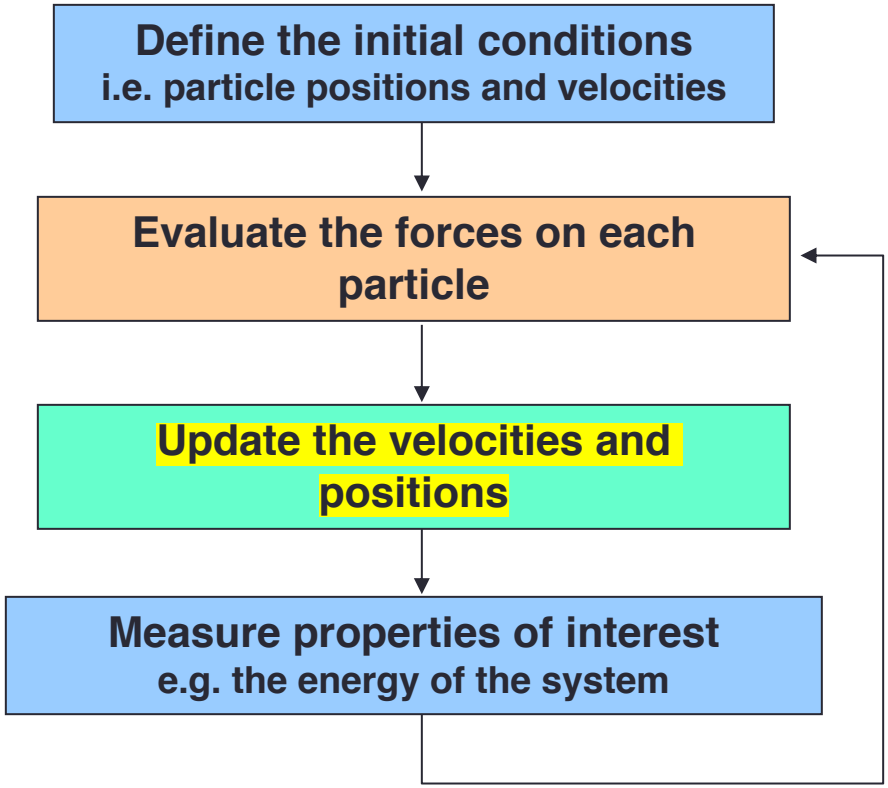
Evaluating the total force on each particle

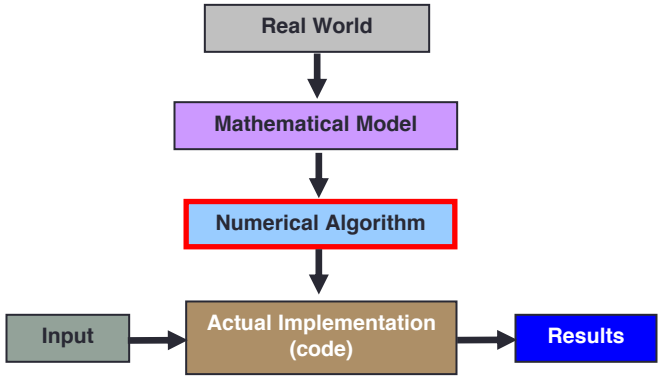
- For each particle, numbered *i* from 1 to *N*
  - add up all forces acting upon that particle due to all other particles.
    - when using periodic boundary conditions, we usually use the nearest image of the other particles.



  - particles should not interact directly with (images of) themselves

General N-body simulation algorithm





epcc

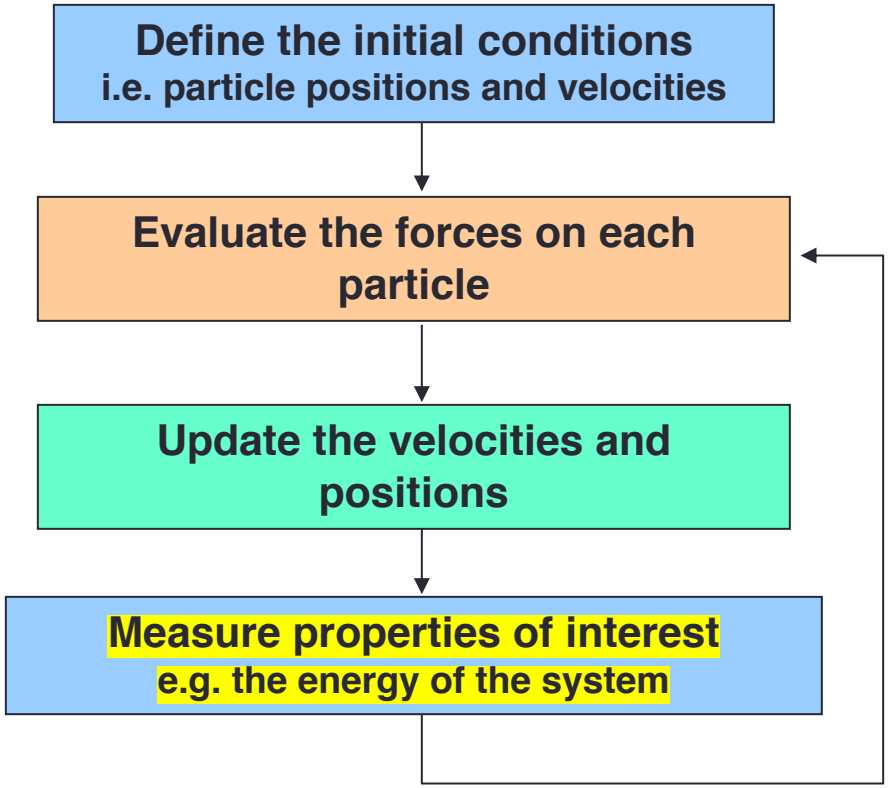
THE UNIVERSITY OF EDINBURGH

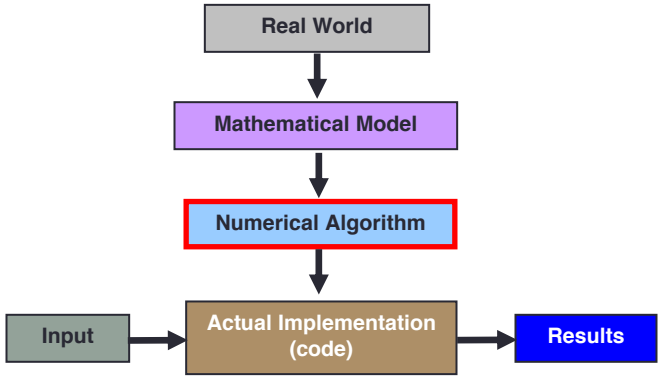
Updating Positions and Velocities

- Update positions and velocities using the discretised forms of Newton's equations of motion and the forces on each particle computed in the previous step

- Many possible ways of doing this
  - Discretisation and time integration schemes trade off speed and accuracy in different ways
  - Consider some of these methods in next lecture & in practical

General N-body simulation algorithm





epcc

THE UNIVERSITY OF EDINBURGH

Measuring properties of interest

- Kinetic energy: $$E_K(t) = \frac{1}{2}\sum_i m_i \vec{v}_i(t)^2$$
- Potential energy: $$E_P(t) = \sum_{\langle i,j \rangle} U(|\vec{r}_j(t) - \vec{r}_i(t)|)$$
  - ($\sum_{\langle i,j \rangle}$ means sum over all pairs)
- Also relevant when we consider errors and accuracy

epcc

THE UNIVERSITY OF EDINBURGH

Computational Cost

Computational cost of force calculations

If each person in this (virtual) room of N people interacts with each other person ("pair-wise"), how many conversations are happening in total?

Computational cost of force calculations

If each person in this (virtual) room of N people interacts with each other person ("pair-wise"), how many conversations are happening in total?

= number of Combinations of 2 people (a pair) chosen from N

Computational cost of force calculations

If each person in this (virtual) room of N people interacts with each other person ("pair-wise"), how many conversations are happening in total?

= number of Combinations of 2 people (a pair) chosen from N

$$= \binom{N}{2} = \frac{N!}{2!(N-2)!}$$

Computational cost of force calculations

Suppose each pairwise force calculation requires execution of **q** operations

(mathematical operations or machine instructions)

Computational cost of force calculations

Suppose each pairwise force calculation requires execution of **q** operations

(mathematical operations or machine instructions)

Suppose each operation takes on average time **t**

Computational cost of force calculations

Suppose each pairwise force calculation requires execution of **q** operations

(mathematical operations or machine instructions)

Suppose each operation takes on average time **t**

Time cost **T** of performing all force calculations is then:

$$T = \binom{N}{2} qt$$

Asymptotic analysis of computational cost

Big O notation

Asymptotic analysis of computational cost

- Want to understand how the runtime of an algorithm scales as a key parameter changes (typically grows)
- Estimating this dependence for very large parameter values: ***asymptotic analysis***
- Asymptotic analysis allows us to
  - Predict which parts of an algorithm dominate runtime in different scenarios
  - Compare the expected relative performance of different algorithms

Big O notation

- Big-O notation specifies upper bound on the quantity of interest (typically runtime, could be memory requirements):

f(N) = O(g(N)) ➔ *for some sufficiently large N*
*it is true that f(N) ≤ c*g(N), where c is a constant*

- e.g. T = $O(N^3)$ implies that for large enough N the runtime scales at most (i.e. no worse than) as N^3

- In practice the value of **c** can be crucial: algorithm that scales worse asymptotically can, for small N, still be much faster

Computational cost of force calculations

- Without writing any code, could perform asymptotic analysis of force calculation runtime based on $T = \binom{N}{2} qt$

Since $\binom{N}{2} = \frac{N!}{2!(N-2)!} = \frac{N(N-1)}{2} = \frac{N^2}{2} - \frac{N}{2}$

and $\lim_{N\to\infty} \binom{N}{2} = \lim_{N\to\infty} \frac{N^2}{2}$

we can conclude that $T = O(N^2)$

Computational cost of force calculations

For N-body simulation the naive serial calculation is

```
for (i=0;i<N;i++){
  for (j=0;j<N;j++){
    if (i != j) {
      force[i] += force(i,j);
    }
  }
}
```

This scales as $O(N^2)$

Asymptotic analysis of computational cost

- Consider: how does the overall runtime of the following sequential matrix-matrix multiplication **A** x **B** scale as a function of the linear size **N** of two square matrices **A**, **B**?

```
for (i=0; i<N; i++){
  for (j=0; j<N; j++){
    for (k=0; k<N; k++){
        c[i][j] += a[i][k]*b[k][j];
    }
  }
}
```

Summary

- N-body simulations compute motion of particles interacting through forces
  - Need to store particle positions and velocities, compute forces, evolve equation(s) of motion
- Evaluating total forces on all atoms is an $O(N^2)$ calculation
  - Asymptotic scaling of computational cost: number of particles doubles ➔ runtime quadruples – this is why force calculation is typically the most time-consuming part of an N-body simulation
- Next lecture (and planetary orbits practical):
  - Discretisation / time integration schemes
  - Accuracy and stability
  - N-body methods: force calculation schemes