Parallel N-body simulations

Toni Collis
EPCC
toni.collis@ed.ac.uk

Outline

- Recap of Lecture 12 introduction to N-body methods
- Review MD algorithm
- Integration schemes
  - Calculating the error

1. Methodology
2. Parallelisation (an introduction)

The Dynamics

- Given the forces,
  - we can determine the acceleration of each particle
  - to update the velocity and position

$$\vec{v}(t+\delta t) = \vec{v}(t) + \vec{a}(t) \times \delta t$$

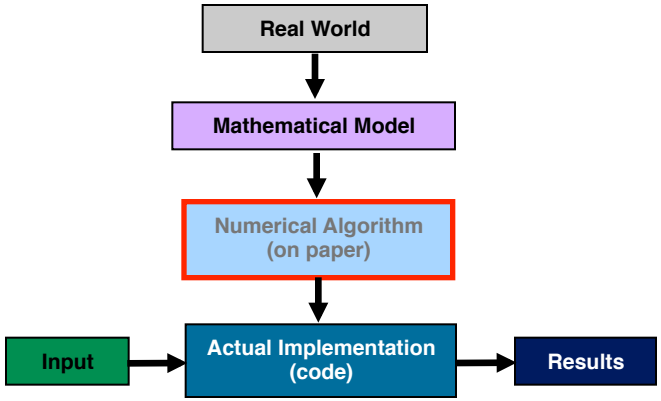
$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t) \times \delta t$$

$$\vec{a}_i = \frac{\vec{F}_i}{m_i}$$

$$\vec{a}_i = \frac{d\vec{v}_i}{dt}$$

$$\vec{v}_i = \frac{d\vec{x}_i}{dt}$$

- Small incremental step = discretisation of the problem
- Then start all over again.



Taylor's Expansion

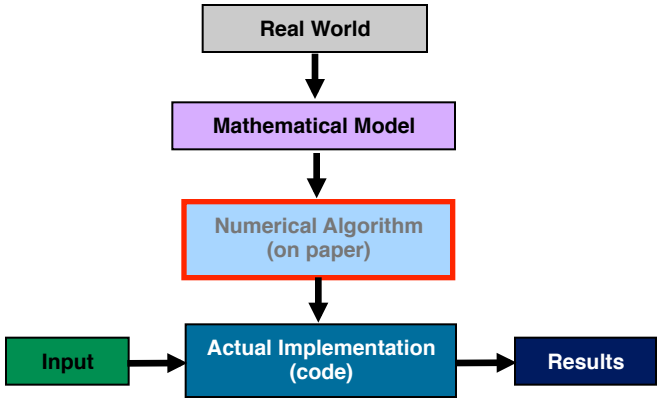
Non-examinable mathematics

- Express a function at a single point as a infinite series of its derivatives evaluated at that point.

$$f(x+h) = \sum_{n=0}^{\infty} \frac{d^n f(x)}{dx^n} \frac{h^n}{n!}$$

Brook Taylor
1685-1731

- Re-express dynamical equations using Taylor's Expansion
- Truncate to some chosen order



The Euler scheme – forward difference

- The simplest is Euler integration:
- Consider Taylor expansion to first order

$$f(x+h) = f(x) + f'(x)h + \mathcal{O}(h^2)$$

$$f'(x) \equiv f^{(1)} \equiv \frac{df(x)}{dx}$$

Ignore terms of this power (and higher)

Leonhard Euler
1707-1783

$$x \rightarrow t \quad h \rightarrow \Delta t$$
$$f'(x) \rightarrow \frac{f(t)}{dt}$$

differential of displacement is velocity
differential of velocity is acceleration

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \vec{a}(t) \times \Delta t$$
$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t) \times \Delta t$$

Euler equations

$$f(x+h) = f(x) + f\prime(x)h + f\prime\prime(x)h^3 + \mathcal{O}(h^4)$$

$$\begin{aligned}
\vec{v}(t+\delta t) &= \vec{v}(t) + \vec{v}\prime(t) \times \delta t + \vec{v}\prime\prime(t) \times \delta t^2 + \mathcal{O}(t^4) \\
&\sim \vec{v}(t) + \vec{v}\prime(t) \times \delta t + \mathcal{O}(t^3) \\
&= \vec{v}(t) + \vec{a}(t) \times \delta t + \mathcal{O}(t^3)
\end{aligned}$$

$$\begin{aligned}
\vec{x}(t+\delta t) &= \vec{x}(t) + \vec{x}\prime(t) \times \delta t + \vec{x}\prime\prime(t) \times \delta t^2 + \mathcal{O}(t^4) \\
&\sim \vec{x}(t) + \vec{x}\prime(t) \times \delta t + \mathcal{O}(t^3) \\
&= \vec{x}(t) + \vec{v}(t) \times \delta t + \mathcal{O}(t^3)
\end{aligned}$$

$$\vec{v}(t+\delta t) = \vec{v}(t) + \vec{a}(t) \times \delta t$$

$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t) \times \delta t$$

Error

- Using a truncated Taylor Expansion is an approximation.
- How would you quantify the error?

$$f(x+h) = f(x) + f\prime(x)h + f\prime\prime(x)h^3 + \mathcal{O}(h^4)$$

$$\begin{aligned}
\vec{v}(t+\delta t) &= \vec{v}(t) + \vec{v}\prime(t) \times \delta t + \vec{v}\prime\prime(t) \times \delta t^2 + \mathcal{O}(t^4) \\
&\sim \vec{v}(t) + \vec{v}\prime(t) \times \delta t + \mathcal{O}(t^3) \\
&= \vec{v}(t) + \vec{a}(t) \times \delta t + \mathcal{O}(t^3)
\end{aligned}$$

$$\begin{aligned}
\vec{x}(t+\delta t) &= \vec{x}(t) + \vec{x}\prime(t) \times \delta t + \vec{x}\prime\prime(t) \times \delta t^2 + \mathcal{O}(t^4) \\
&\sim \vec{x}(t) + \vec{x}\prime(t) \times \delta t + \mathcal{O}(t^3) \\
&= \vec{x}(t) + \vec{v}(t) \times \delta t + \mathcal{O}(t^3)
\end{aligned}$$

Euler and the error

$$\vec{v}(t+\delta t) = \vec{v}(t) + \vec{a}(t) \times \delta t$$

$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t) \times \delta t$$

- Finite difference equation looks very similar to the differential equation
  - What is the error?
- Truncated Taylor series at $\mathcal{O}(h^2) \sim \mathcal{O}(t^2)$
- Evaluate *v* and *r* at time τ in *N* steps of size $\delta(t) = \frac{1}{N}$
- The error for each step is $\mathcal{O}((t^2))$
- Total error is $N \times (\delta t)^2 = \frac{1}{\delta t}(\delta t)^2 = \delta t$

Error is linear in step size

Verlet – second order scheme

- A better, more stable and symmetric scheme
- Combines forward and backward Taylor expansions

Loup Verlet
b. 1931

$$\vec{x}(t+\delta t) = 2\vec{x}(t) - \vec{x}(t-\delta t) + \vec{a}(t) \times (\delta t)^2$$

The velocities do not appear explicitly, but can be found using $\vec{v}(t+\delta t) = \dfrac{x(t+\delta t) - r(t-\delta t)}{2\delta t}$

Discretisation errors:

Positions are correct to $\mathcal{O}((\delta t)^4)$

Velocities are correct only to $\mathcal{O}((\delta t)^2)$

This algorithm is not self starting

Requires one Euler step to be performed first

Other forms exist to address these shortcomings:

Leapfrog and Velocity Verlet

Leapfrog – second order scheme

- Uses half-step velocities:

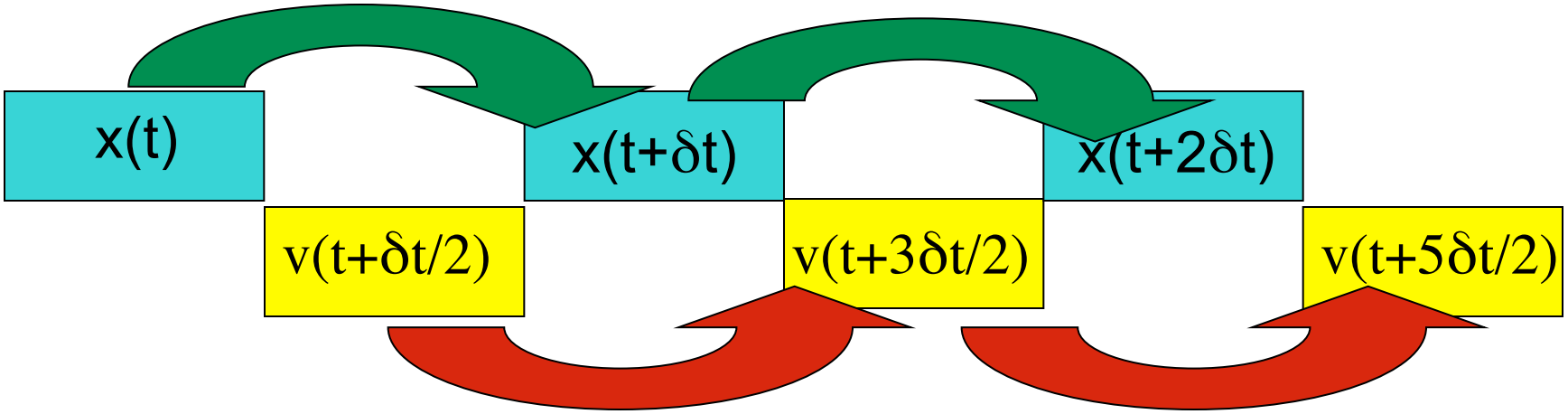
$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t+1/2\delta t) \times \delta t$$

$$\vec{v}(t+1/2\delta t) = \vec{v}(t-1/2\delta t) + \vec{a}(t) \times \delta t$$

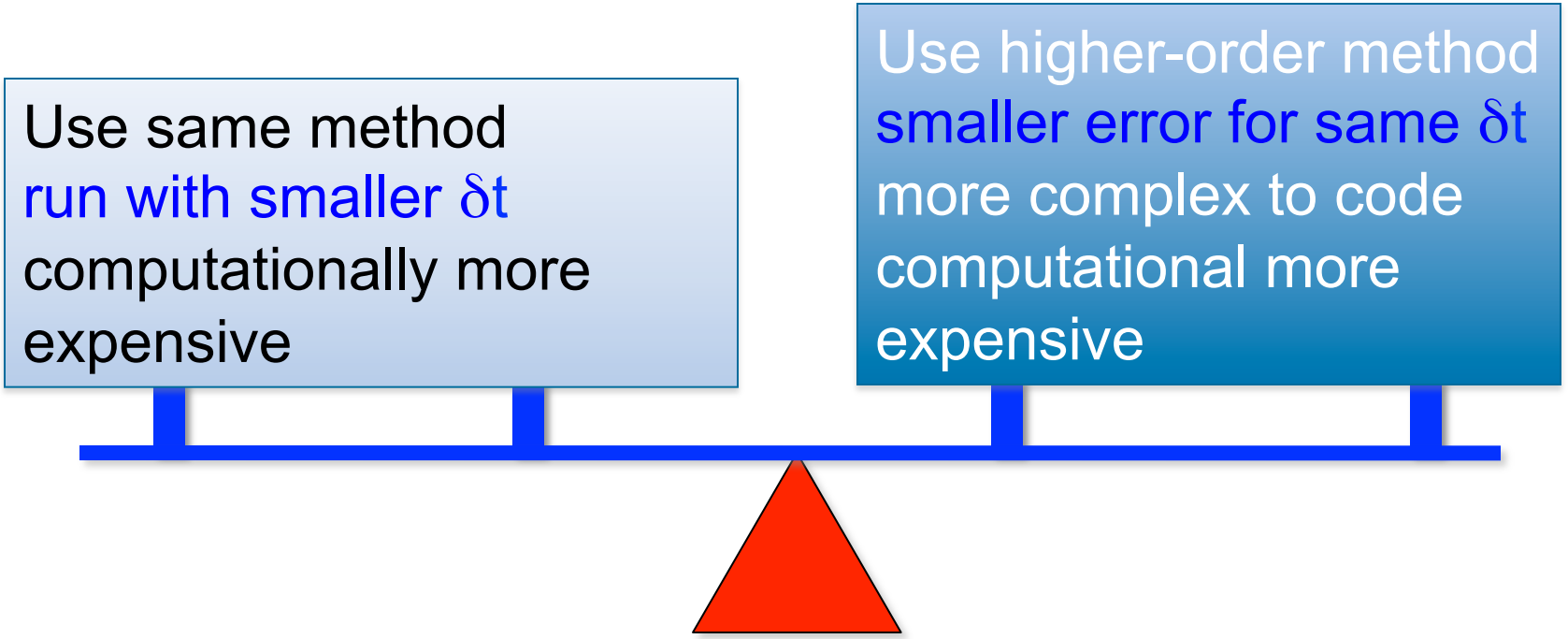
- Combines forward and backward 2nd order Taylor expansion
  - both displacement and velocity
- Error for each step is $\mathcal{O}((\delta t)^3)$

Algorithm is not self-starting: Require Euler 1st step

N steps of size δt -> Total error is O((δ)^2))



Reducing the error



- A huge variety of schemes exist
  - some may even be adaptive and choose Δt automatically
  - beyond the scope of this course
  - Earth's 2nd moon calculation uses high order integrator scheme

Question

What are the other sources of error in a simulation?

- Algorithm (Verlet etc.)
- Measurements of experiment that simulation is being compared with
- Inaccurate theory (e.g. Newtonian mechanics)
- Time-step size
- Limitations of artificial boxsize

Checking Correctness of Solution

- This is a real issue!

- May be able to monitor special quantities
  - energy must be conserved
  - momentum must be conserved
    - useful checks but not really enough
- Qualitative answer should not depend on step size
  - Multiple simulations to confirm answer

Checks & Sums

- Conservation of energy

$$E_K(t) = \sum_i \frac{1}{2} m_i \vec{v}_i(t)$$

$$E_P(t) = \sum_{\langle i,j \rangle} U(|\vec{r}_j(t) - \vec{r}_i(t)|)$$

$$E_T = E_P(t) + E_K(t) = C$$

Constant: Not time dependant

Similarly Conservation of linear momentum

$$\vec{P} = \sum_i m_i \vec{v}_i(t)$$

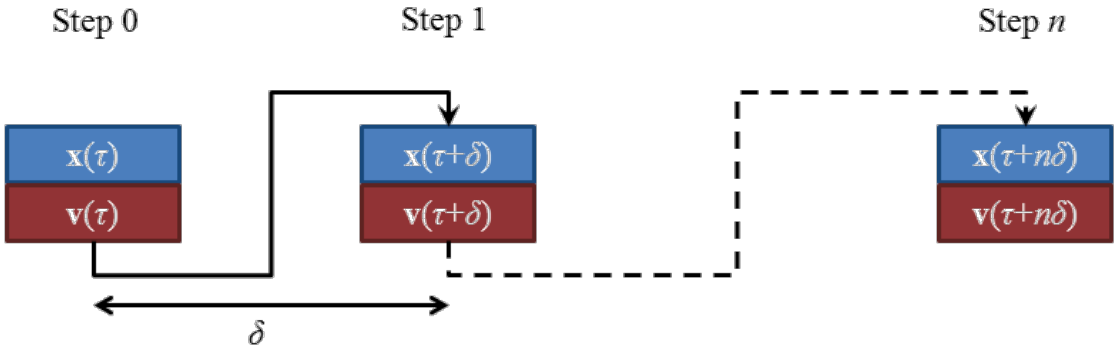
Linear momentum should also be conserved in each dimension individually, e.g. *x, y, z*

Recap

- N-body simulations approximate the motion of particles by treating a particle as a point in space
- Calculate net force on an atom (takes into consideration other particles and any external forces) – $O(N^2)$ calculation
- Asymptotic computational time: as number of particles doubles, runtime quadruples

Recap: Time integration examples

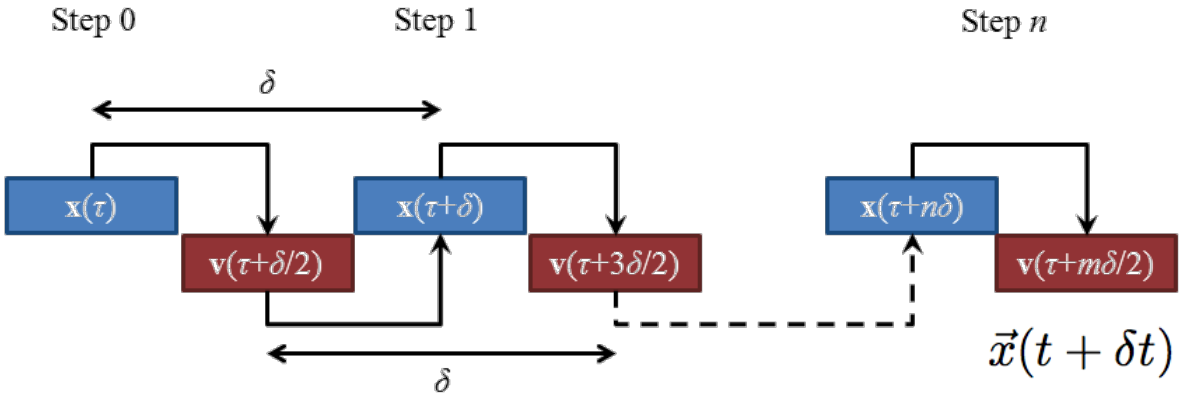
- Euler scheme



$$\vec{v}(t+\delta t) = \vec{v}(t) + \vec{a}(t) \times \delta t$$

$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t) \times \delta t$$

- Leapfrog scheme



$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t+{}^1\!/\!_2\delta t) \times \delta t$$

$$\vec{v}(t+{}^1\!/\!_2\delta t) = \vec{v}(t-{}^1\!/\!_2\delta t) + \vec{a}(t) \times \delta t$$

Methods for calculating the N-body force

The 'Direct Method'

- Brute force approach to solving the N-Body problem:
  - calculate at each time step the force on each object as the sum of the forces from all other objects.
- Simplification: forces are all equal and opposite, and do not need to be calculated twice.
- For i = 1 to n
  - Force(i)=0
- For i = 1 to n
  - For j=i+1 to n
    - Force(i) += force between i and j
    - Force(j) -= force between i and j

Potential and Force

- In particle simulations often use ‘potential’ to describe particle interactions
- Potential energy is energy that results from position or configuration
- The force on an object is the negative of the derivative of the potential energy *P:*

$$F = - dP/dt$$

- For gravitational forces:

$$F = GMm/r^2$$

$$P = -GMm/r$$

Particle radii

- Point particles: technically treating particles as ‘points’ allows them to coincide
- Forces are typically 1/r (or $1/r^{2\cdots}$): there is a singularity in the force as particles get very close to each other.
- Time-stepping method: these large forces are calculated as if they last the entire timestep, often resulting in an overestimation of the total force effect. Overcome this by having a very small time-step, but then need to recalculate all forces for each small time-step

Particle radii

- Smaller timesteps and multi-step methods can be used, but ultimately the steep nature of the potential (and hence force) is that there is a limit to how accurately close encounters can be calculated
- Use a ‘fudge factor’ to maintain stability (albeit at the loss of accuracy) in the simulation.
- Simple approach: all particles are collisionless spheres of some radius, for which each interaction is treated as a point mass if distances are greater than and zero if distances are less than the closest approach radius.

What is the alternative?

Particle radii

- Potential (and hence force) functions can be modified to include a softened distance, effectively treating all distances as if they were some small distance greater than they actually are.
- For a potential *P*, treat all distances as if they were some small distance *ε* greater than the real separation *r. eg:*

$$P = -G\frac{Mm}{r} \rightarrow P = -G\frac{Mm}{(r^2+\epsilon^2)^{1/2}}$$

$$\vec{F}_i = G\frac{M_i m_j(\vec{x}_i - \vec{x}_j)}{x_i - x_j}$$

- This introduces only a small error

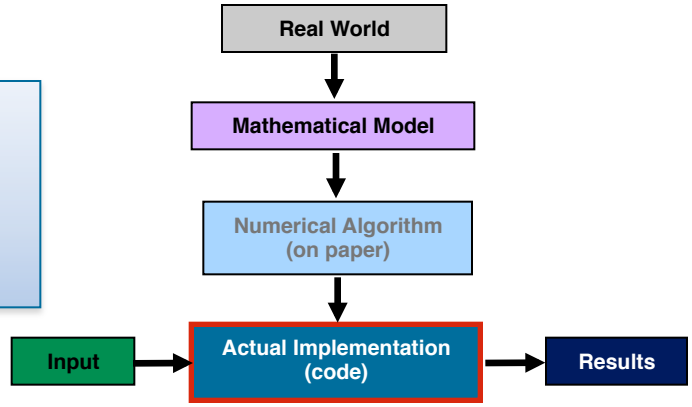
Parallelisation

- The problem
  - Each atom is treated as a point mass
  - Easily split space into domains, with atoms located in a domain and assigned to a particular process
  - Every single pair of atoms interacts (even across domains)
  - Atoms move: do they move domains?
- Solutions?
  - Atom decomposition
  - Force based decomposition
  - Spatial decomposition
- First: what general parallelisation techniques are available to us?

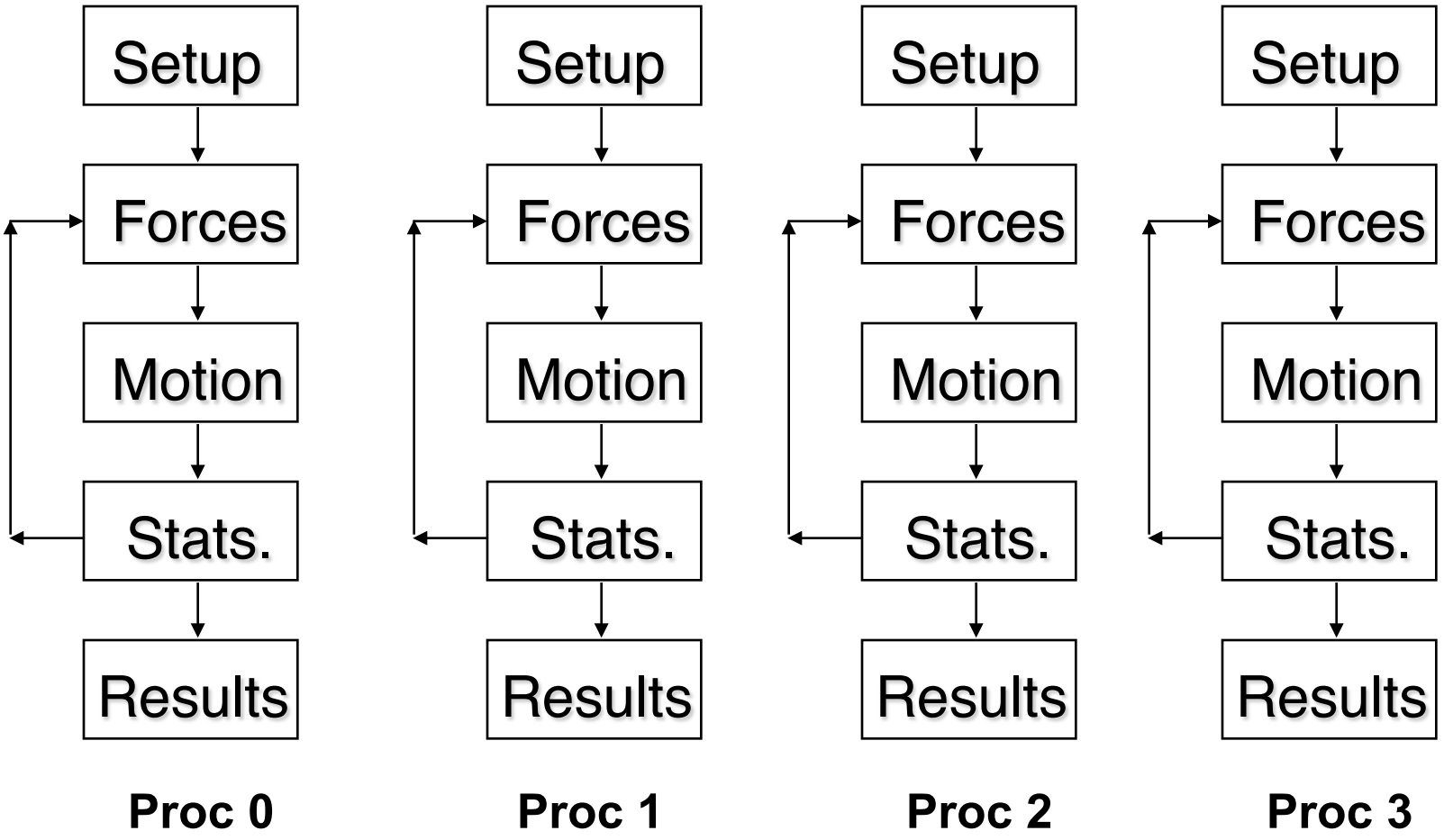
Parallelisation strategies

1. Simple task farm
2. Replicated data
3. Systolic loops
4. Domain decomposition – most common on MPP machines
5. Dynamic task allocation

- Other approaches are possible but uncommon

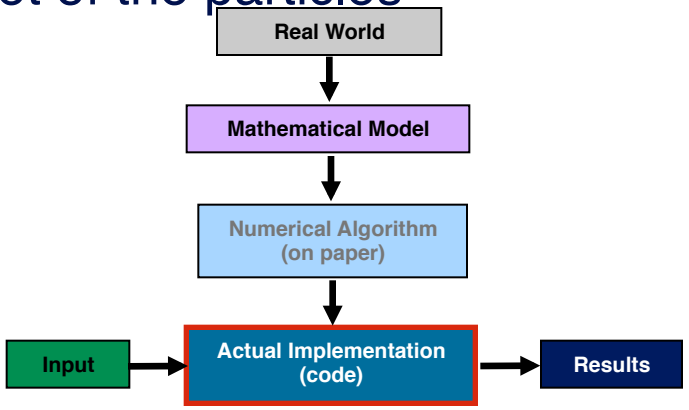


Simple task farm

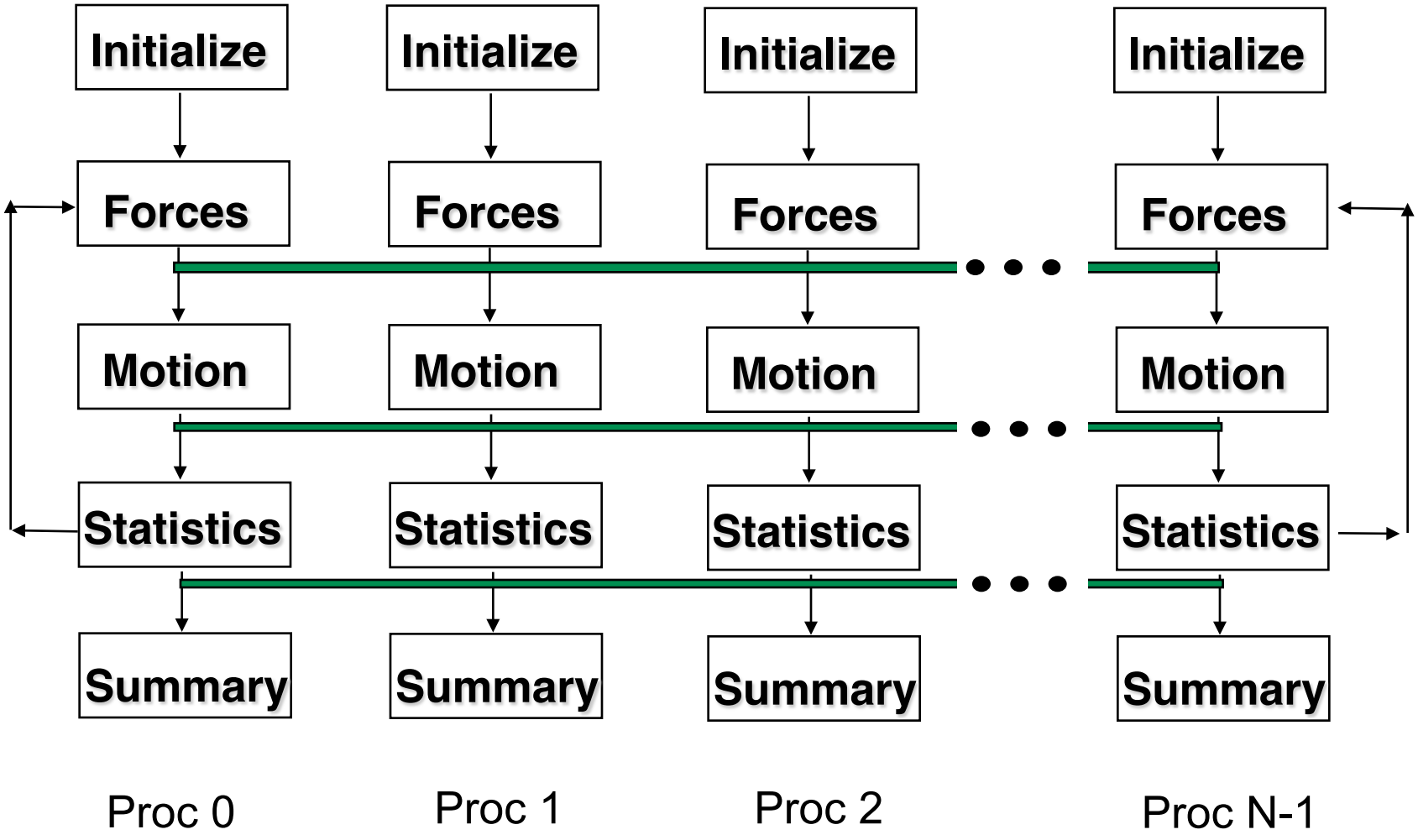


Simple task farm

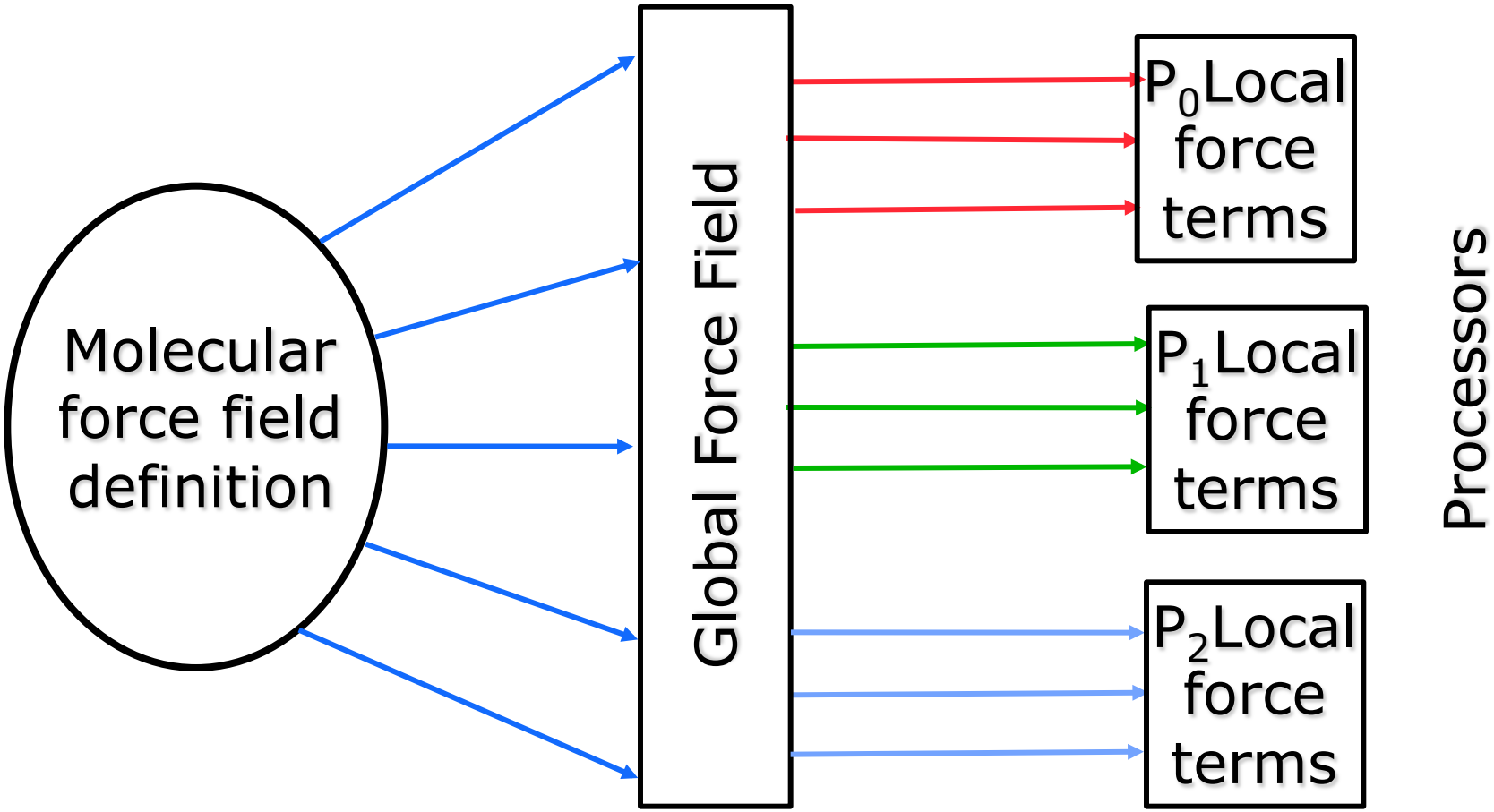
- Advantages:
  - Simple to implement - no communications
  - Maximum parallel efficiency – excellent throughput
  - Perfect load balancing
  - Good scaling behaviour
  - Suitable method for stochastic and replica simulations
- Disadvantages:
  - Limited to short timescale dynamics
  - For traditional MD tasks can just be a subset of the particles calculations



Replicated data



Replicated data

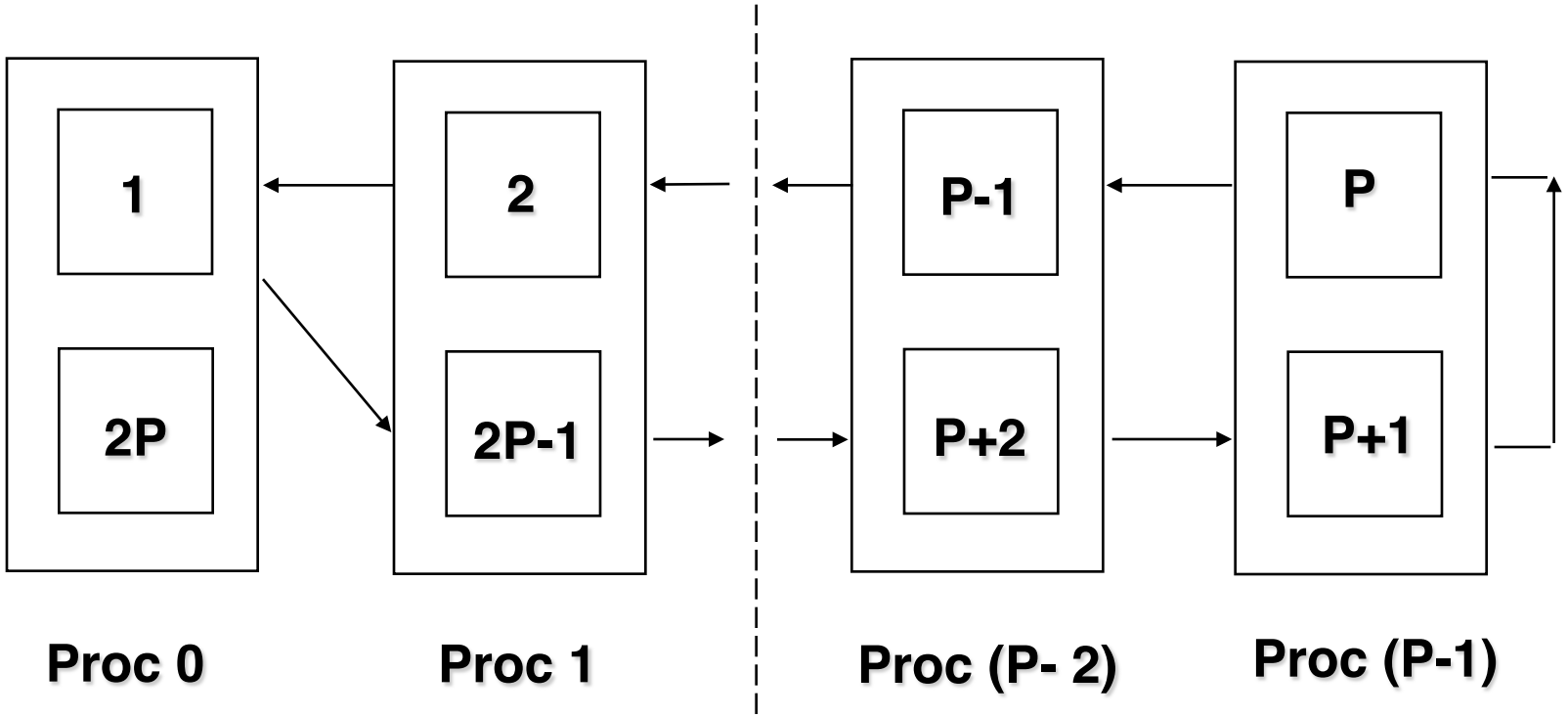


Replicated data

- Advantages:
  - Simple to implement
  - Good load balancing
  - Highly portable programs
  - Suitable for complex force fields
  - Good scaling with system size
  - Dynamic load balancing possible
- Disadvantages:
  - High communication overhead
  - Sub-optimal scaling with processor count
  - Large memory requirement
  - Unsuitable for massive parallelism

Systolic loop algorithms

- Compute the interactions between (and within) “data packets”
- Data packets are then transferred between nodes to permit calculation of all possible pair interactions

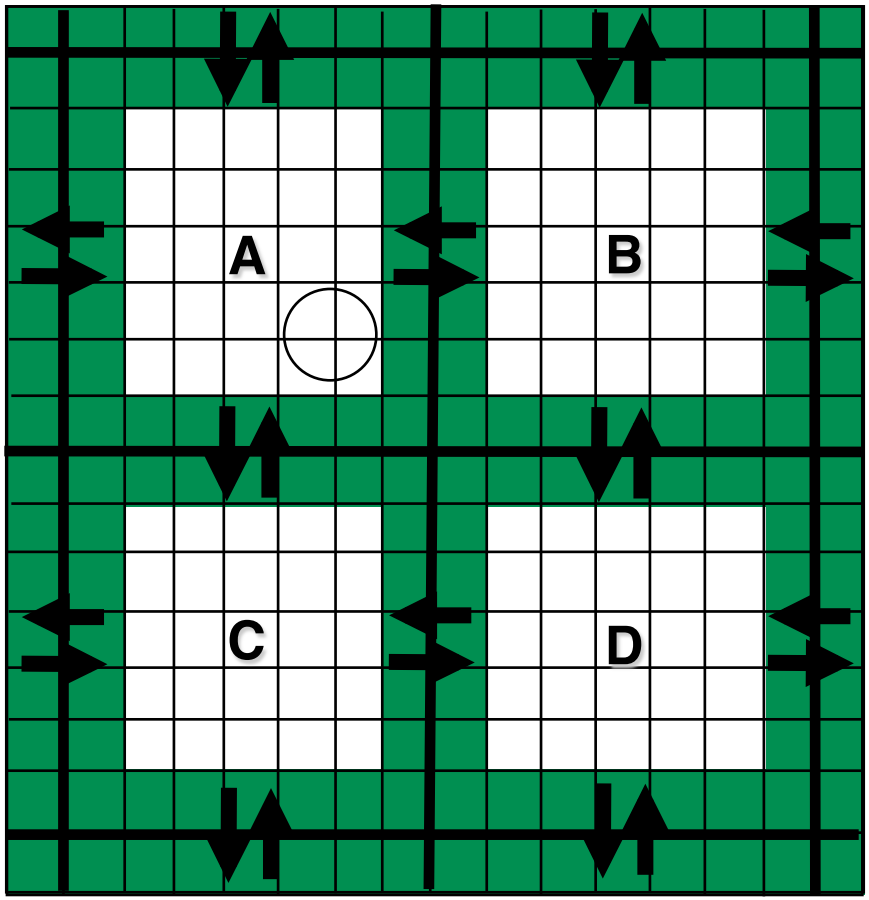


Systolic loops

- Advantages
  - Good load balancing
  - Portable between parallel machines
  - Good scaling with system size and processor count
  - Memory requirement fully distributed
  - Asynchronous communications
- Disadvantages
  - Complicated communications strategy
  - Complicated (n-body) force fields difficult

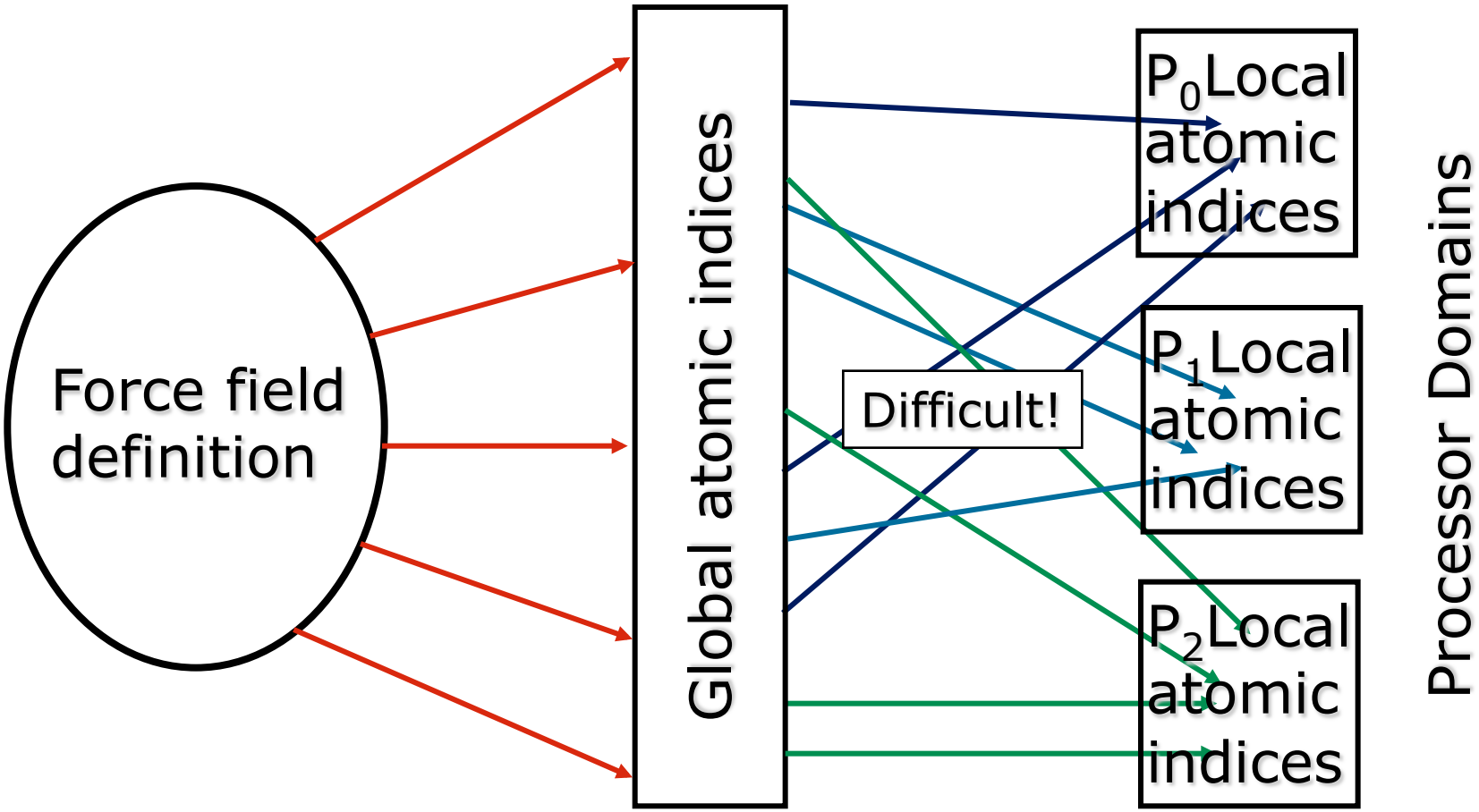
Domain decomposition

2D Example



- Short range potential cut off ($r_{cut} << L_{cell}$)
- Spatial decomposition of atoms into domains
- Map domains onto processors
- Use *link cells* in each domain
- Pass border link cells to adjacent processors
- Calculate forces, solve equations of motion
- Re-allocate atoms leaving domains

Domain decomposition



Domain decomposition

- Advantages:
  - Predominantly local communications
  - Good load balancing (if system is isotropic!)
  - Ideal for huge systems (~ 10^5 atoms or more)
  - Good scaling with system size
  - Fully distributed memory requirement
  - Dynamic load balancing possible
  - Predominantly local communications
- Disadvantages
  - Problems with mapping/portability
  - Sub-optimal scaling with processors for smaller systems
  - Requires relatively short potential cut off
  - Complex (n-body) force fields tricky
  - Can be a very complicated communication structure

Summary

- Algorithm: initialise, calculate separation, calculate forces, update positions and velocities, repeat.
- Integration schemes
  - Calculating the error
- Methodology
- Parallelisation (an introduction)

Orbits

- Simulates the orbit of a single planet around a single star (the sun)
  - the position of the star remains fixed
  - this is a one-body problem, for which exact solutions exist
  - allows us easily to evaluate the accuracy of each scheme
- Code available in FORTRAN or C
- See practical sheet for full details

Practical

- Implement and compare two simplest integration schemes
  - Euler and Leapfrog
- Play with different orbits
  - compare with known analytic result
- Explore the dynamics
  - How to measure the error
  - What does effect does the step-size have
  - What effect does the scheme have