AST 1420

Galactic Structure and Dynamics

Q&A

Reminders

- Assignment 2 posted
- Proposed presentation topic due today

Agenda

- N-body gravity solvers
- Orbit integration
- N-body simulations

N-body solvers

N-body solvers

- Problem faced in all *N*-body simulations: mutual gravity between *N* bodies

$$\Phi(\mathbf{x}_j) = -G \sum_{i\neq j} \frac{m_i}{|\mathbf{x}_j - \mathbf{x}_i|} \, .$$

- $1/|x_j-x_i|$ can be softened, but basic problem stands:
    - Naive implementations: $O(N)$ / particle x N particles —> $O(N^2)$
- Collisional simulations require high-precision gravity —> directly compute the sum
- Collisionless simulations can smooth the gravitational field and use this to speed up code

Softening

- In collisionless *N*-body simulations, particles are typically *softened*
- Replace $$\Phi(\mathbf{x}_j) = -G \sum_{i \neq j} \frac{m_i}{|\mathbf{x}_j - \mathbf{x}_i|} \,.$$
  with $$\Phi(\mathbf{x}_j) = G \sum_{i \neq j} m_i \, S(|\mathbf{x}_j - \mathbf{x}_i|)$$
- Simple, common softening: point-mass —> Plummer sphere
$$S(r; \varepsilon) = -\frac{1}{\sqrt{r^2 + \varepsilon^2}}$$
- Softening removes the unphysical force divergence when ‘particles’ get close
- Doesn’t reduce two-body relaxation much though, just close encounters

Softening

- Do particles get close enough together that softening is necessary?
  - Remember relaxation time: t_{relax} ~ N / log N x t_{dyn} —> all N-body simulations of galaxies have N << Ntrue —> too many interactions between particles
    - Choose N such that $t_{relax} >> t_{sim}$
    - But typically still leaves some non-physical interactions, including at $b < b_{min}$ (b_{min} is where we cut off the interactions in the calculation of the relaxation time)

Softening kernel

$$\Phi(\mathbf{x}_j) = -G \sum_{i \neq j} \frac{m_i}{|\mathbf{x}_j - \mathbf{x}_i|} \, . \quad \longrightarrow \quad \Phi(\mathbf{x}_j) = G \sum_{i \neq j} m_i \, S(|\mathbf{x}_j - \mathbf{x}_i|)$$

- Replace 1/r potential with softened form that does not diverge as r —> 0 and has the same total mass

- For example, Plummer

$$S(r; \varepsilon) = -\frac{1}{\sqrt{r^2 + \varepsilon^2}}$$

- Softening length epsilon: to prevent large-angle deflections in system of size R

$$\epsilon_{\mathrm{2body}} \sim 2R/N$$

- Maximum inter-particle force < typical mean field

$$\epsilon_{\mathrm{min}} \sim R/\sqrt{N}.$$

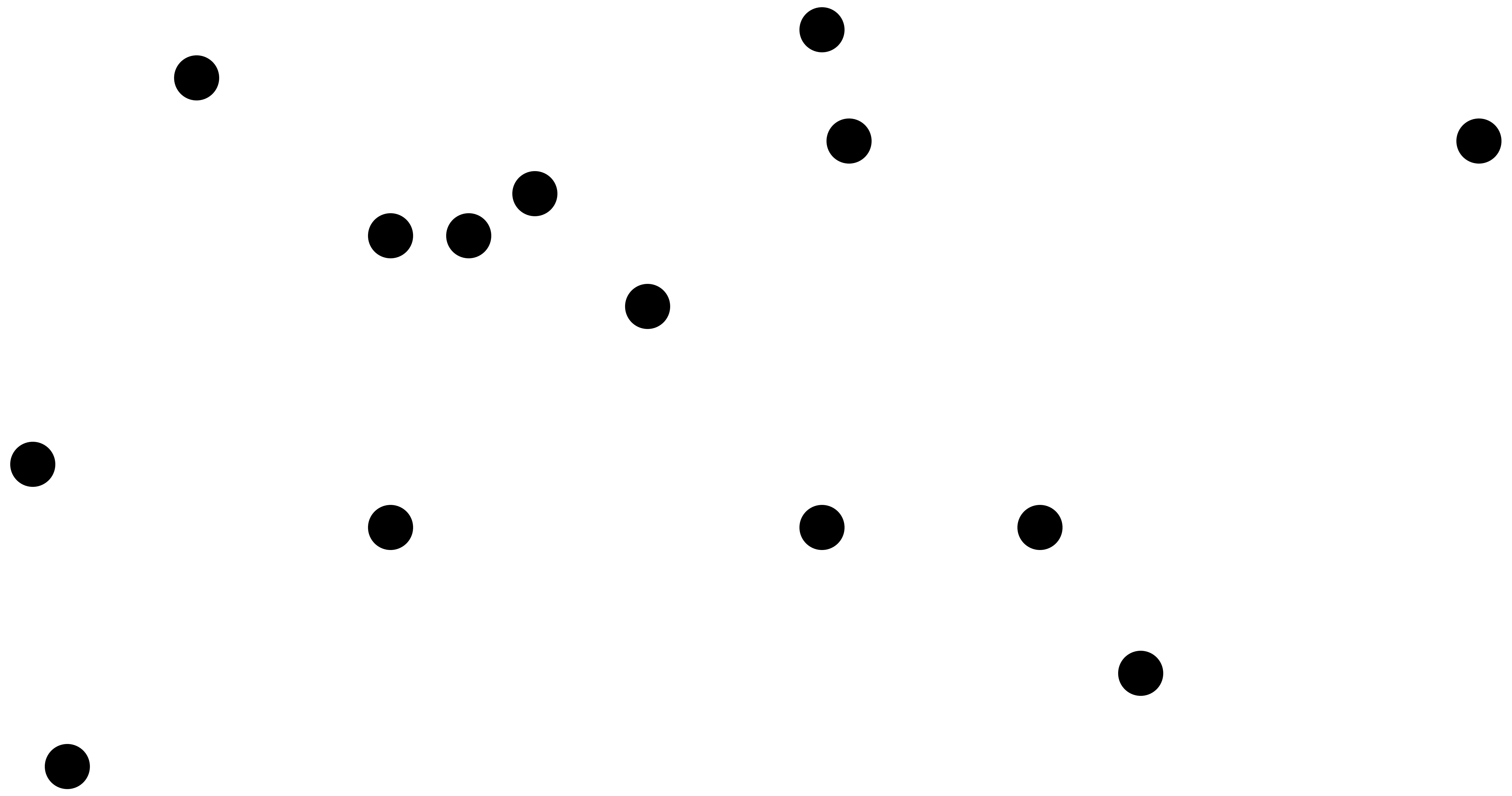
Softening kernel in a tree

$$\Phi(\mathbf{x}_j) = -G \sum_{i \neq j} \frac{m_i}{|\mathbf{x}_j - \mathbf{x}_i|} \, . \longrightarrow \Phi(\mathbf{x}_j) = G \sum_{i \neq j} m_i \, S(|\mathbf{x}_j - \mathbf{x}_i|)$$

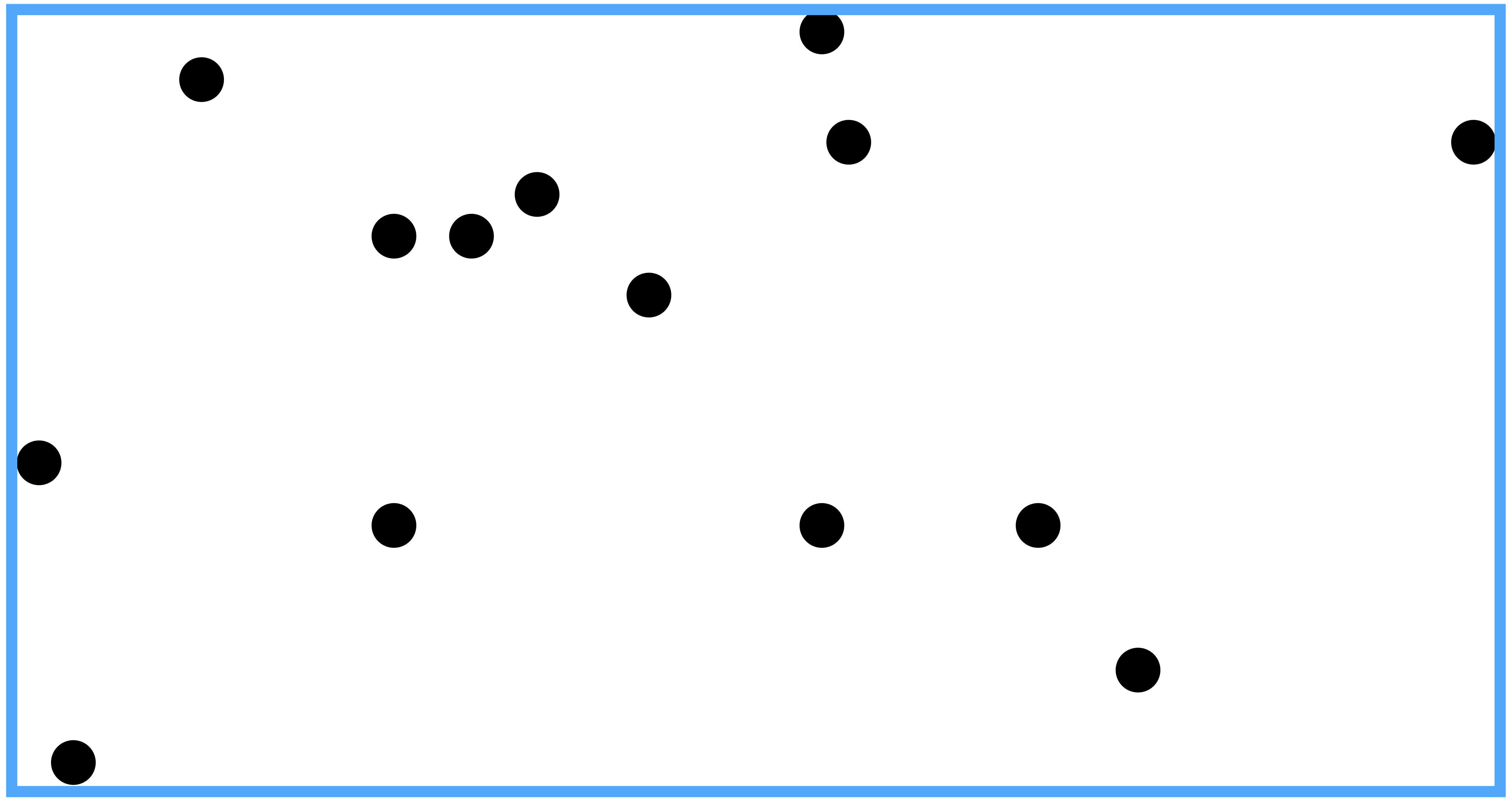
- In tree implementation, we calculate the potential by expanding the potential around each tree cell's center of mass
- For smoothening, need to expand the potential coming from the softening kernel rather than 1/r

$$\Phi_C(\mathbf{x}_j) \approx G \sum_{|\mathbf{n}| \leq p} M_{\mathbf{n}}(\bar{\mathbf{x}}_C) \, \nabla^{\mathbf{n}} \, S(\mathbf{x}_j - \bar{\mathbf{x}}_C)$$

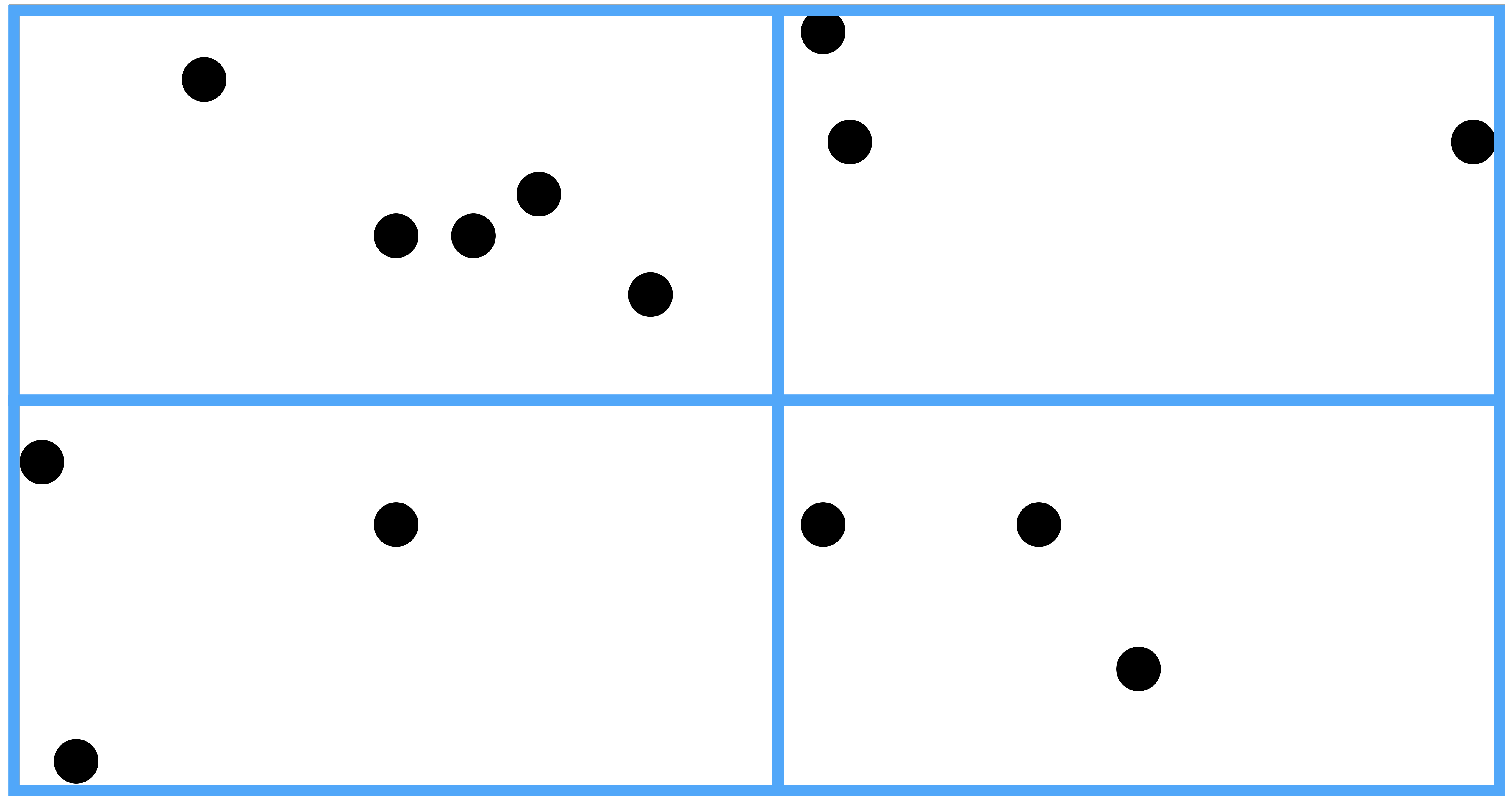
Hierarchical tree



Hierarchical tree

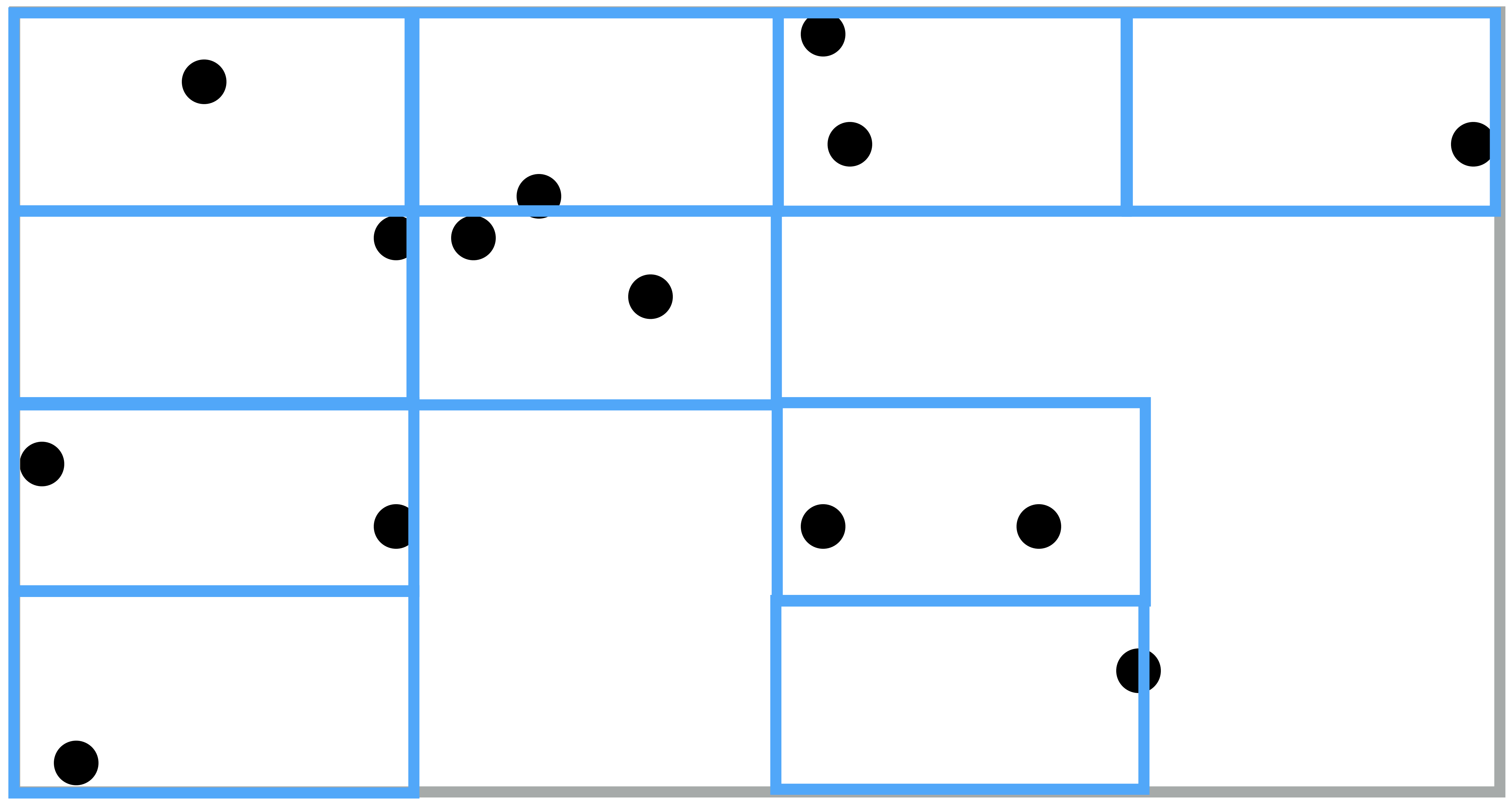


Hierarchical tree



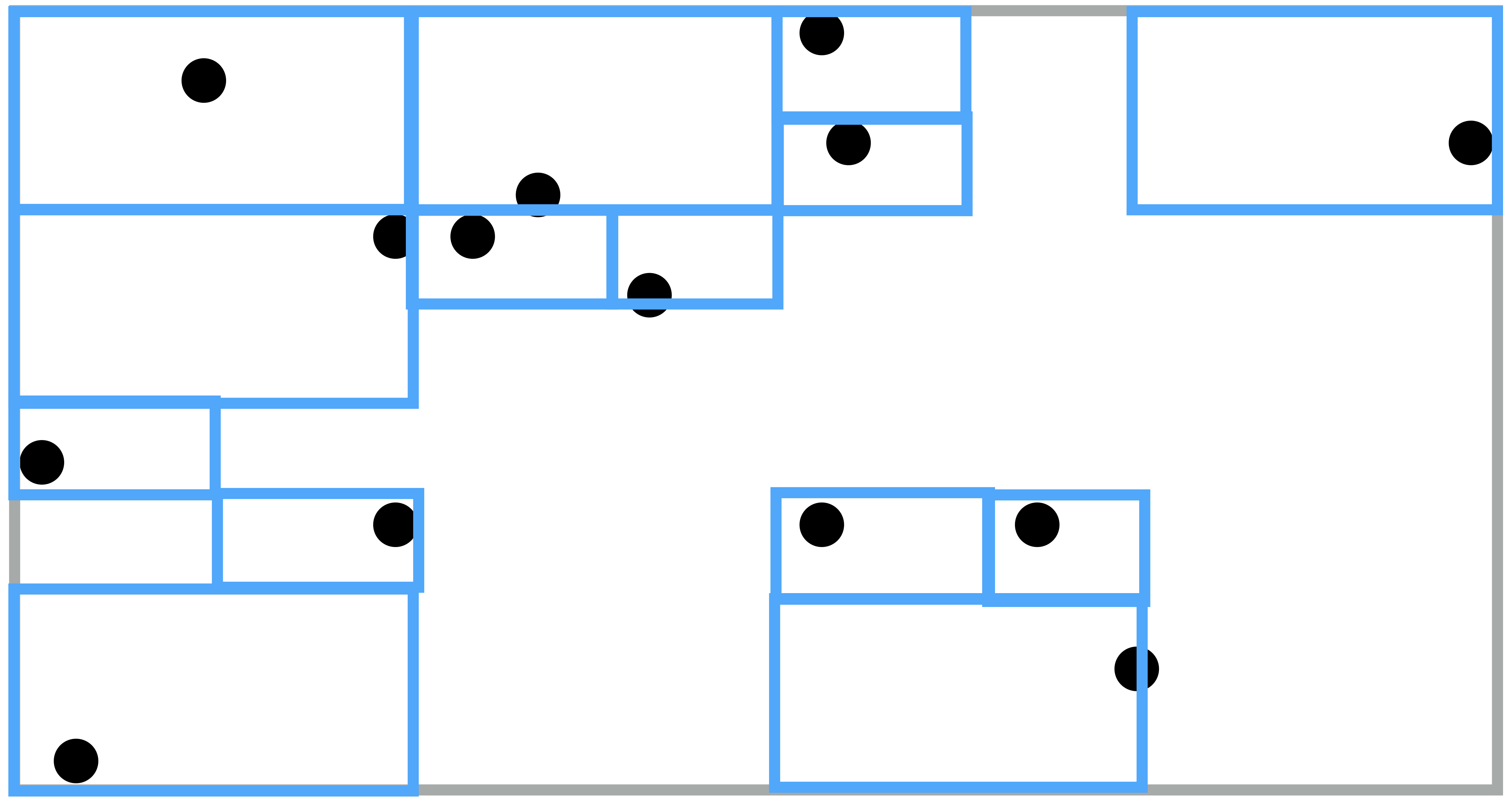
Always divide in half

Hierarchical tree



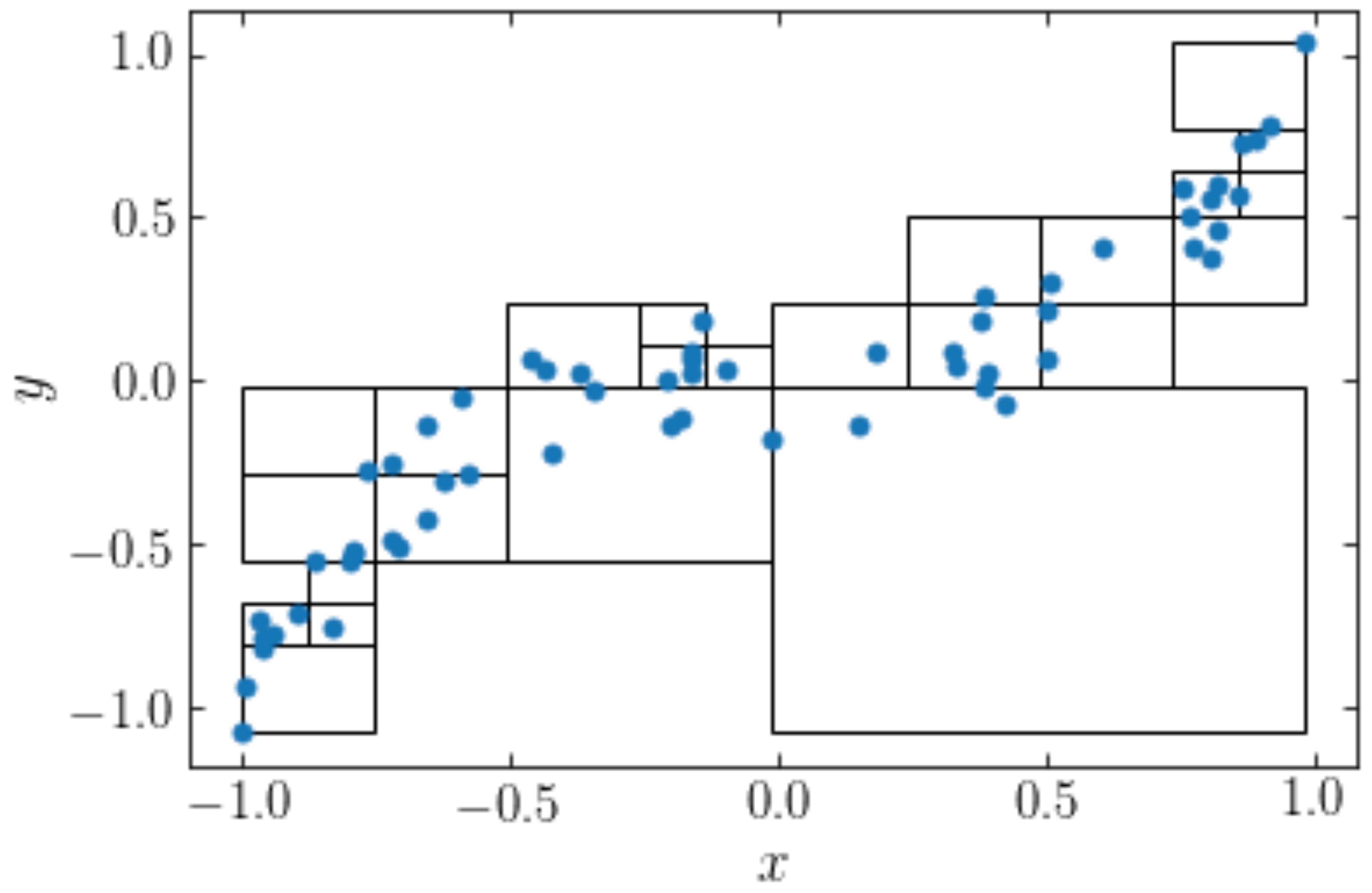
Always divide in half

Hierarchical tree



Always divide in half

Hierarchical tree: real quad-tree example

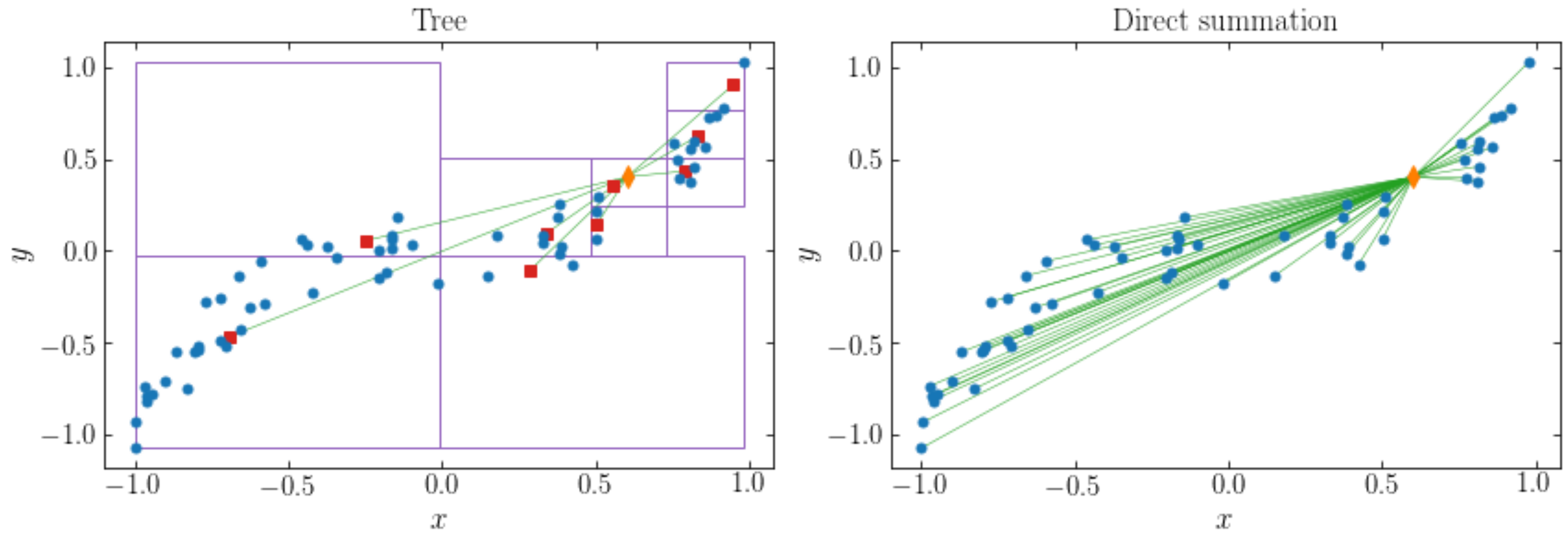


Stop when $N_{\text{in cell}} < N_{\text{limit}}$

Set N_{limit} based on computational considerations

Gravity approximation

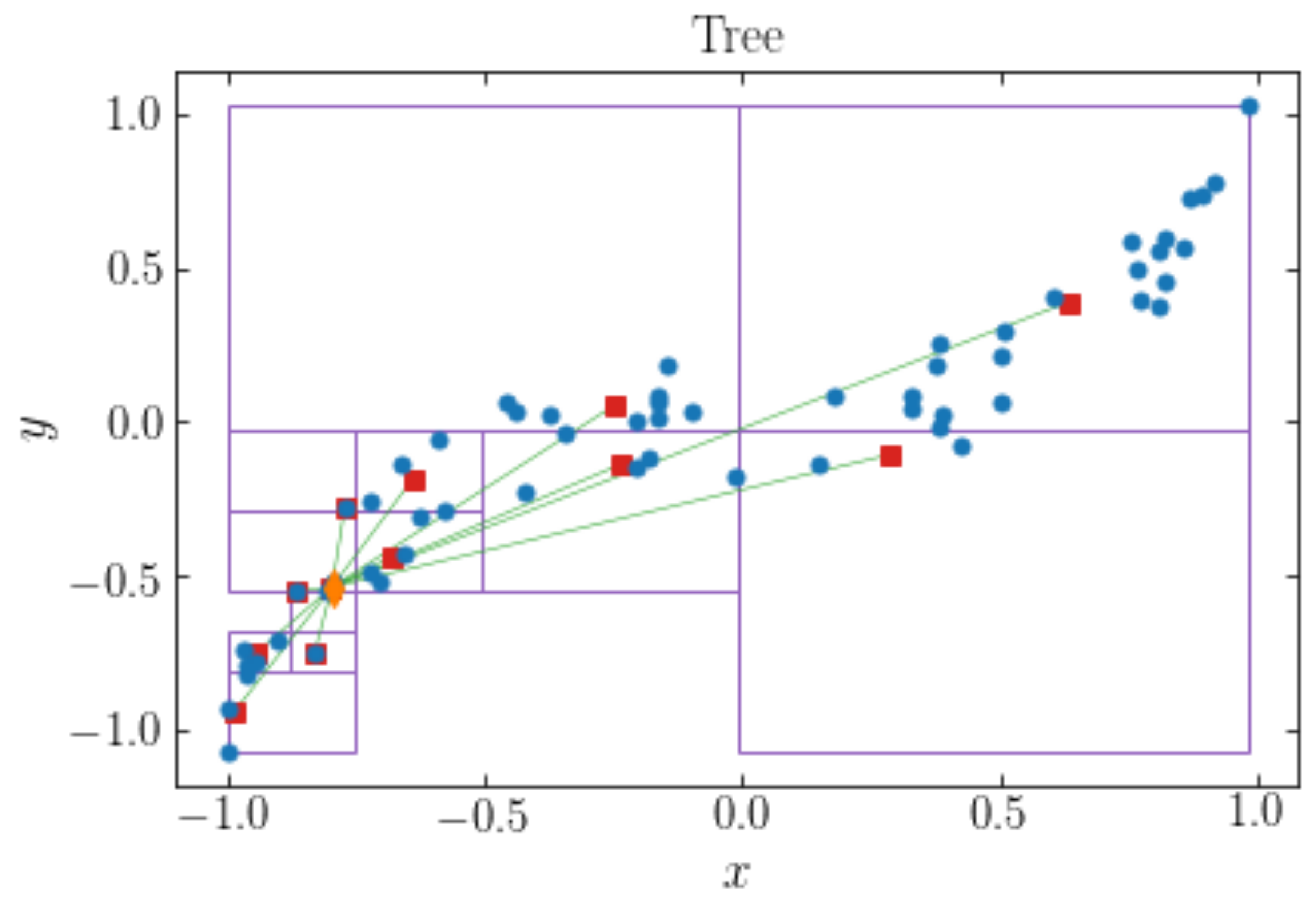
$\theta_{\text{lim}} = 0.5$

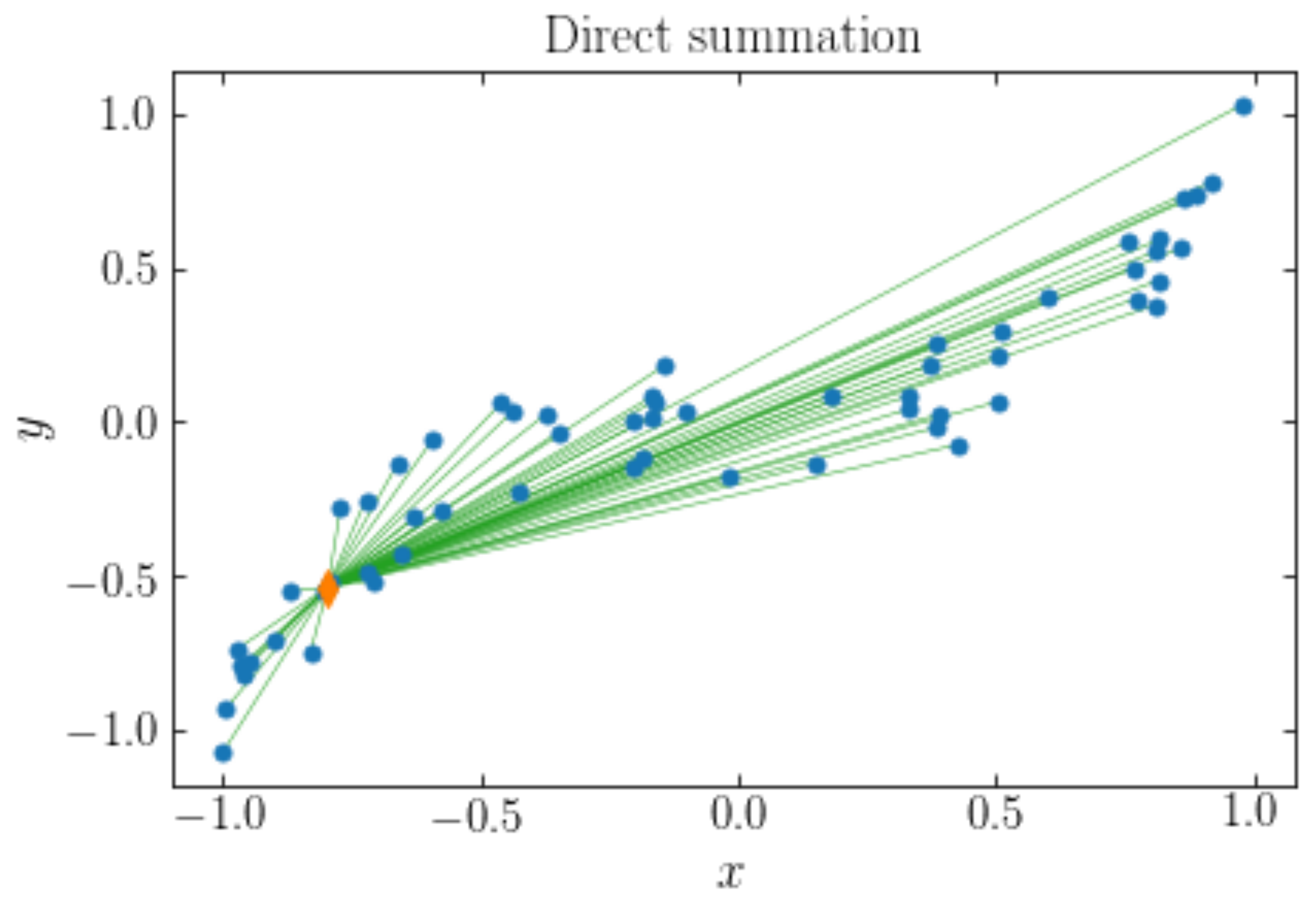


Expand around center of mass, not the center

Gravity approximation

$\theta_{\text{lim}} = 0.5$





Expand around center of mass, not the center

(Re)-building the tree

- Tree structure changes at each time step —> need to re-calculate the tree at every time step
  - Tree set up is O(N log N) and typically quite fast, so it's not *that* bad to re-calculate it…
  - Can keep the structure of the tree the same for K time steps, but update the center of mass and multipole moments based on particle trajectories, only fully re-build tree once it's gone out of date
  - Moving particles between cells is difficult (typically on a different computational node; lots of complicated algorithms to keep nearby particles nearby in memory), so re-build once too many particles are moving across boundaries

Orbit integration

Hamiltonian integration

- Also known as symplectic integration (due to the symplectic nature of Hamiltonian mechanics)
- Discretize the Hamiltonian in the following way

$$\begin{aligned} H(\mathbf{q},\mathbf{p}) &= \frac{|\mathbf{p}|^2}{2} + \Phi(\mathbf{q})\ \text{III}(t;\Delta t) \\ &= \frac{|\mathbf{p}|^2}{2} + \Phi(\mathbf{q})\,\Delta t \sum_{j=-\infty}^{\infty} \delta\,(t - j\,\Delta t) \end{aligned}$$

- Comb is unphysical, simply a computational device!

Hamiltonian integration

- Can also discretize the Hamiltonian in other ways
- For example, in planetary systems, the Hamiltonian is

$$H(\vec{q},\vec{p}) = H_{\mathrm{Sun}}(\vec{q},\vec{p}) + \Phi_{\mathrm{planets}}(\vec{q},\vec{p},t)$$

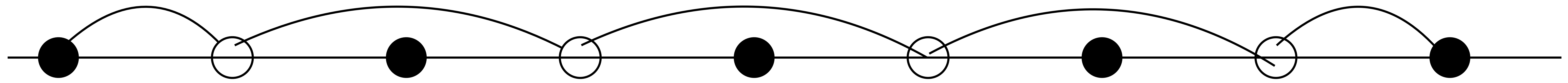
- and we can discretize this as

$$H(\vec{q},\vec{p}) = H_{\mathrm{Sun}}(\vec{q},\vec{p}) + \Phi_{\mathrm{planets}}(\vec{q},\vec{p},t)\,\sum_{j=-\infty}^{\infty}\delta(t-j\,\Delta t)$$

- The ‘leapfrog’ integrator for this becomes:
    - Drift: solve Keplerian orbit
    - Kick: apply forces from other planets

Leapfrog integration

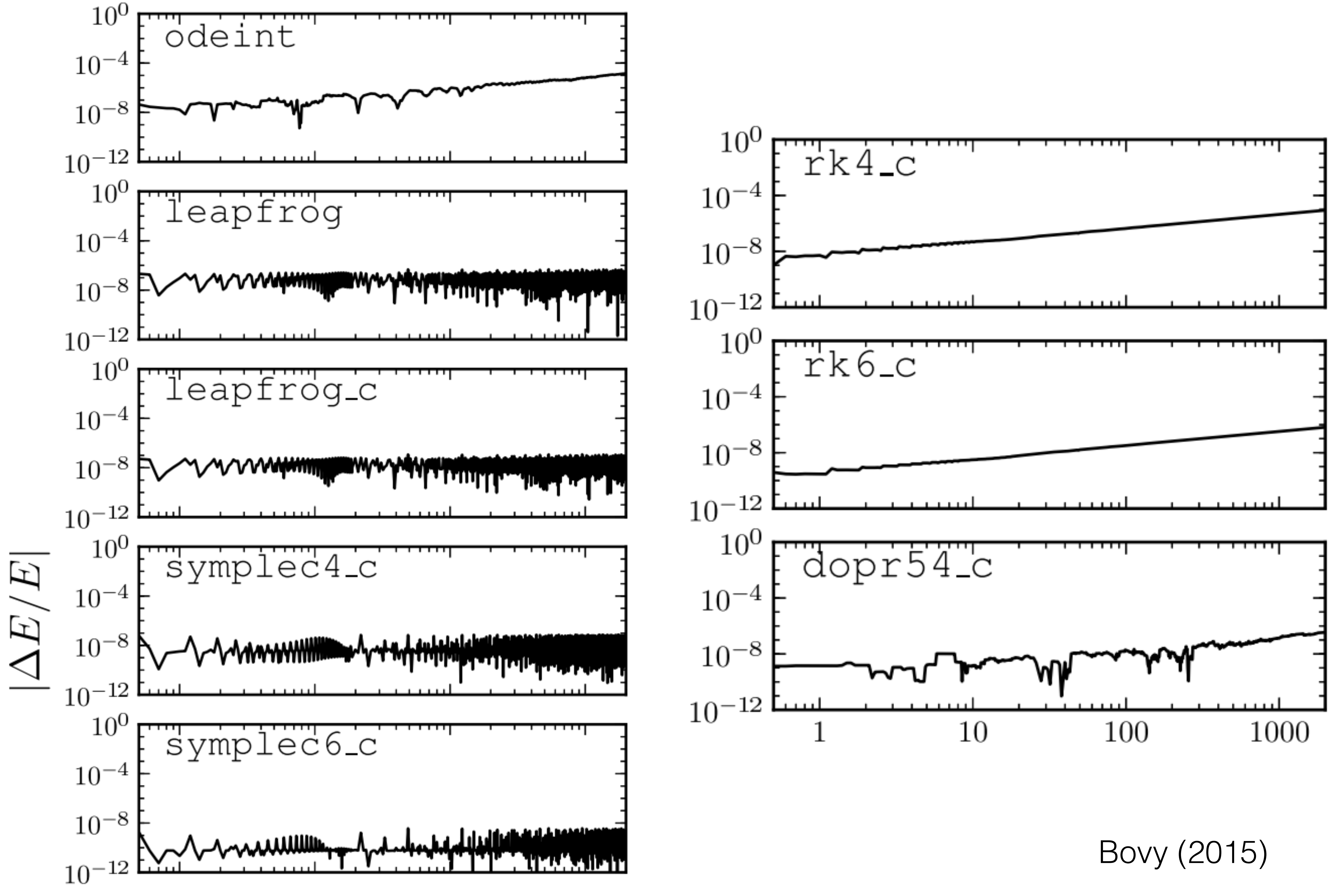
- Leapfrog drift-kick-drift is a second order methods (so errors scale as [Delta t]^3), while kick-drift is a first-order method
- Force evaluation is generally the computationally expensive part and these two integrators require the same number of force evaluations (a single kick)
- Moreover, if we don't need to know the position at the end of the time step, we can combine the second drift of the previous step with the first drift of the second step



Combining Hamiltonian and ‘regular’ orbit integration

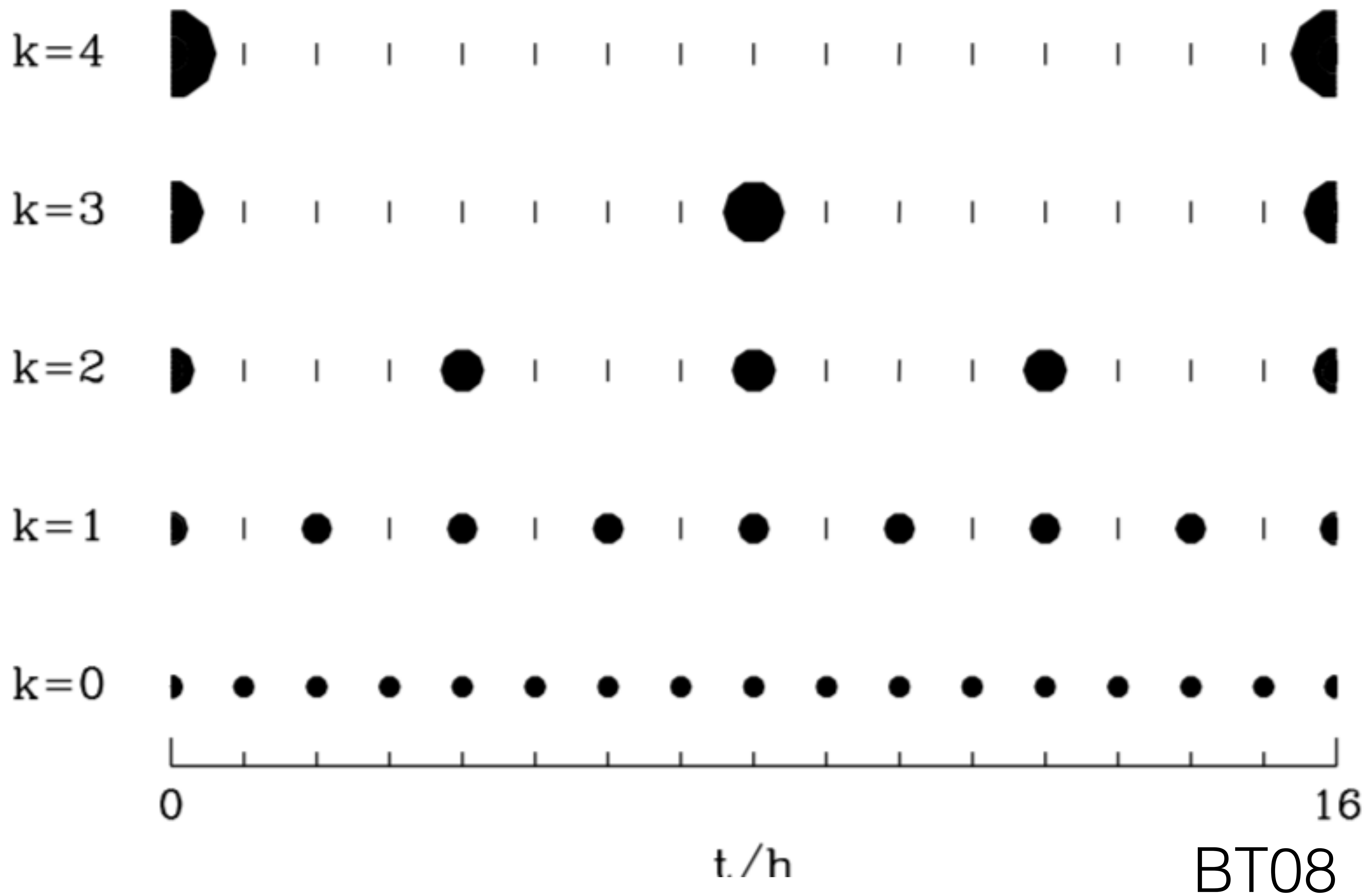
- Advantage of Hamiltonian integration is its long-term stability, but it comes at the cost of getting the short-term behavior wrong
- That’s a problem when interesting things happen on short time scales
- For example, a collision
- Really only a problem in systems that we want to integrate for many many many dynamical times, like planetary systems
- Hybrid integrators detect collisions and then switch to regular, high-order integrators to resolve the collision before going back to the Hamiltonian integrator

Energy conservation



Bovy (2015)

Block time-step scheme



BT08

Block time step scheme

- How do you decide which level to place a particle on?
- Ideally, you would know each particle’s dynamical time
- But this is very difficult to know, because we don’t know in general what a particle is orbiting around (e.g., in a merging galaxy simulation)
- Typically set the timescale for each particle as the time scale on which the acceleration is changing, estimated or computed (based on the jerk)
- E.g.,

$$\Delta t_i = \eta \sqrt{|\Phi_i|}/|\boldsymbol{a}_i|,$$

- Block time steps are used in most big simulations of galaxies

Why does an N-body simulation work?

Collisionless *N*-body modeling

- Coupled equations: collisionless Boltzmann equation

$$\frac{\partial f(\mathbf{x},\mathbf{v},t)}{\partial t} + \mathbf{v}\,\frac{\partial f(\mathbf{x},\mathbf{v},t)}{\partial \mathbf{x}} - \frac{\partial \Phi(\mathbf{x},t)}{\partial \mathbf{x}}\,\frac{\partial f(\mathbf{x},\mathbf{v},t)}{\partial \mathbf{v}} = 0\,.$$

- Plus Poisson

$$\nabla^2 \Phi(\mathbf{x},t) = 4\pi\, G M \int \mathrm{d}\mathbf{v}\, f(\mathbf{x},\mathbf{v},t)$$

- Poisson has formal solution

$$\Phi(\mathbf{x},t) = -G M \int \mathrm{d}\mathbf{x}' \int \mathrm{d}\mathbf{v}'\,\frac{f(\mathbf{x}',\mathbf{v}',t)}{|\mathbf{x}'-\mathbf{x}|}$$

- We know the initial condition f(**x**,**v**,t=0) and want to know how it evolves

Method of characteristics for solving a PDE

- An initial-value PDE can be solved using the method of characteristic
- For example

$$a(x,t)\,\frac{\partial u}{\partial x} + b(x,t)\,\frac{\partial u}{\partial t} + c(x,t)\,u = 0$$

- with u(x,t=0) = f(x_0)
- Change coordinates from (x,t) to (x,s) such that

$$\frac{\mathrm{d}x}{\mathrm{d}s} = a(x,t)$$
$$\frac{\mathrm{d}t}{\mathrm{d}s} = b(x,t)$$

- The PDE then becomes and ODE

$$\frac{\mathrm{d}u}{\mathrm{d}s} + c(x,t) = 0$$

- with initial condition u(0) = f(x0)

Method of characteristics for solving a collisionless N-bod y simulation

- Change coordinates from (**x**,**v**,t) to (**x**,v,s) such that

$$\frac{\mathrm{d}t}{\mathrm{d}s} = 1\,,$$
$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} = \mathbf{v}\,,$$
$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}s} = -\frac{\partial\Phi(\mathbf{x},t)}{}$$

- CBE becomes $\frac{\mathrm{d}f}{\mathrm{d}s} = 0\,.$

- Solved by f(s) = constant = f($\mathbf{x}_0$,$\mathbf{v}_0$)

- Also t=s

- Other two equations are Hamilton's equation

- Characteristic curves are therefore orbits!

Method of characteristics for solving a collisionless N-bod y simulation

- Find the characteristic curves by
    - Monte Carlo sample a set of initial conditions (N initial conditions for characteristic curves)
    - Compute the gravitational potential / forces
    - Solve for the orbits of the initial conditions
    - Potential changes as the orbits evolve, so need to re-compute it along the way
- An N-body simulation is therefore nothing more than solving the CBE with characteristic curves
- The N bodies and their orbits are nothing more than Monte Carlo samples from the distribution function and the characteristic curves of the solution
- *they don’t correspond to physical objects*

Types of simulations we can do

- Tree codes are good for basically any galaxy simulation
- For cosmology, it can be computationally expedient to use a Fourier (particle-mesh) technique for the large scales
- Can we integrate forward observed data?
  - No!
  - Errors are too big
  - N-body simulations are weakly chaotic on ~dynamical time, so while we can trust the overall statistics, the individual trajectories are not accurate