

Astro PG Course

# cosmological simulations

session 1: background and the N-body method

Sownak Bose  
&  
Shaun Brown

[sownak.bose@durham.ac.uk](mailto:sownak.bose@durham.ac.uk)

 @Swnk16



**lecture notes:**

<https://github.com/sownakbose/AstroPGCourse-Nbody>

**jupyter notebooks:**

[https://github.com/Shaun-T-Brown/Summer\\_school](https://github.com/Shaun-T-Brown/Summer_school)  
[link also on lecture notes Github page]



# objectives of this course

- give you some historical context of cosmological simulations and how they aid us understand the universe
- a (brief) introduction to some of the numerical techniques used in modern codes
- how to set up initial conditions and introduce cosmology to an N-body simulations
- algorithms for identifying nonlinear structures (haloes and subhaloes) in a particle distribution
- figure out how to design and run your own simulations
- learn techniques for analysing the outputs of simulations
- appreciate where simulations are useful, but also what their limitations are!

# objectives of this course

- give you some historical context of cosmological simulations and how they aid us understand the universe
- a (brief) introduction to some of the numerical techniques used in modern codes
- how to set up initial conditions and introduce cosmology to an N-body simulations
- algorithms for identifying nonlinear structures (haloes and subhaloes) in a particle distribution
- figure out how to design and run your own simulations
- learn techniques for analysing the outputs of simulations
- appreciate where simulations are useful, but also what their limitations are!

day one

# objectives of this course

- give you some historical context of cosmological simulations and how they aid us understand the universe
- a (brief) introduction to some of the numerical techniques used in modern codes
- how to set up initial conditions and introduce cosmology to an N-body simulations
- algorithms for identifying nonlinear structures (haloes and subhaloes) in a particle distribution
- figure out how to design and run your own simulations
- learn techniques for analysing the outputs of simulations
- appreciate where simulations are useful, but also what their limitations are!

day one

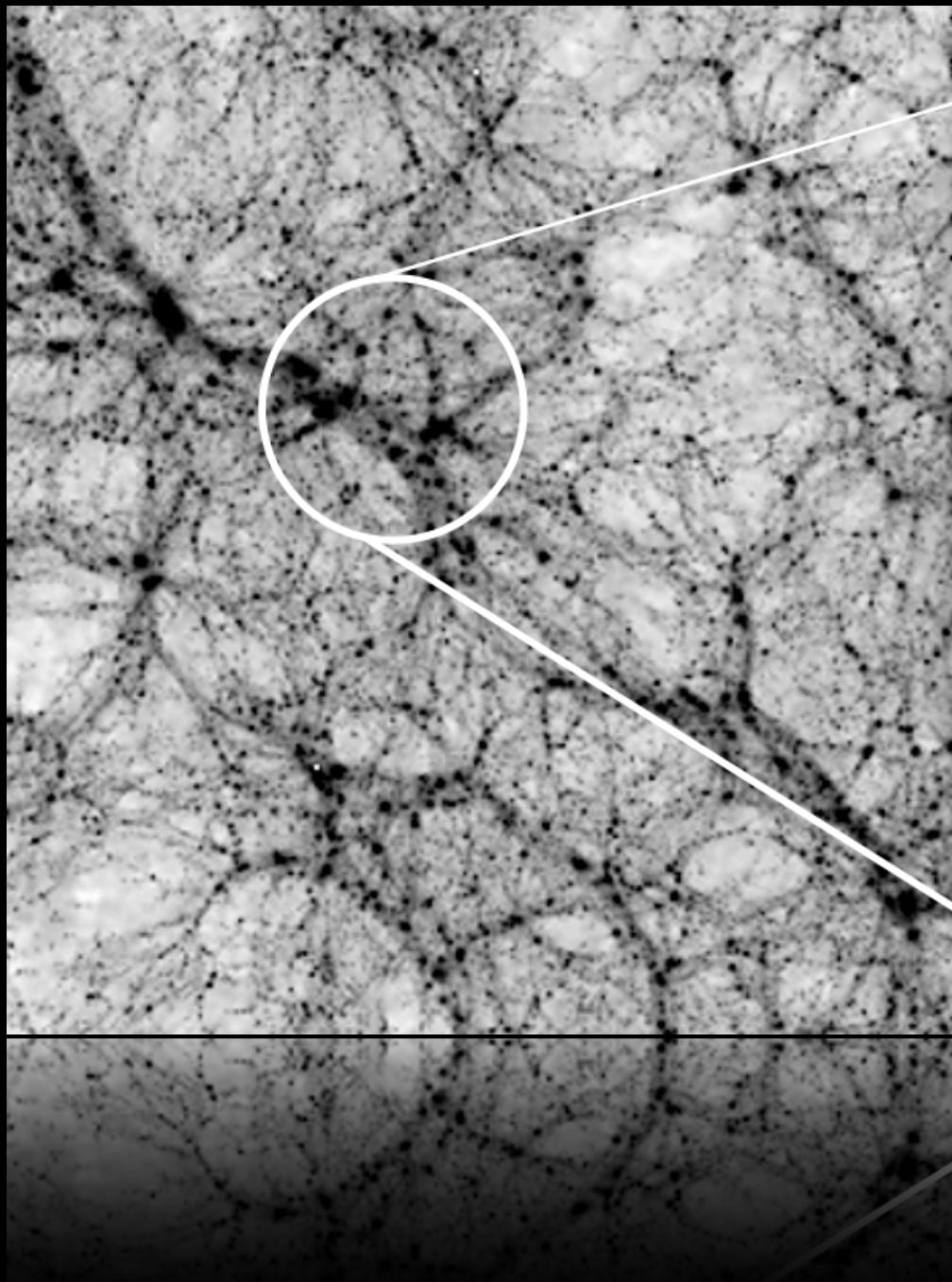
day two

# objectives of this course

- give you some historical context of cosmological simulations and how they aid us understand the universe | day one
- a (brief) introduction to some of the numerical techniques used in modern codes |
- how to set up initial conditions and introduce cosmology to an N-body simulations | day two
- algorithms for identifying nonlinear structures (haloes and subhaloes) in a particle distribution | day three
- figure out how to design and run your own simulations
- learn techniques for analysing the outputs of simulations
- appreciate where simulations are useful, but also what their limitations are!

# structures are coupled across multiple scales

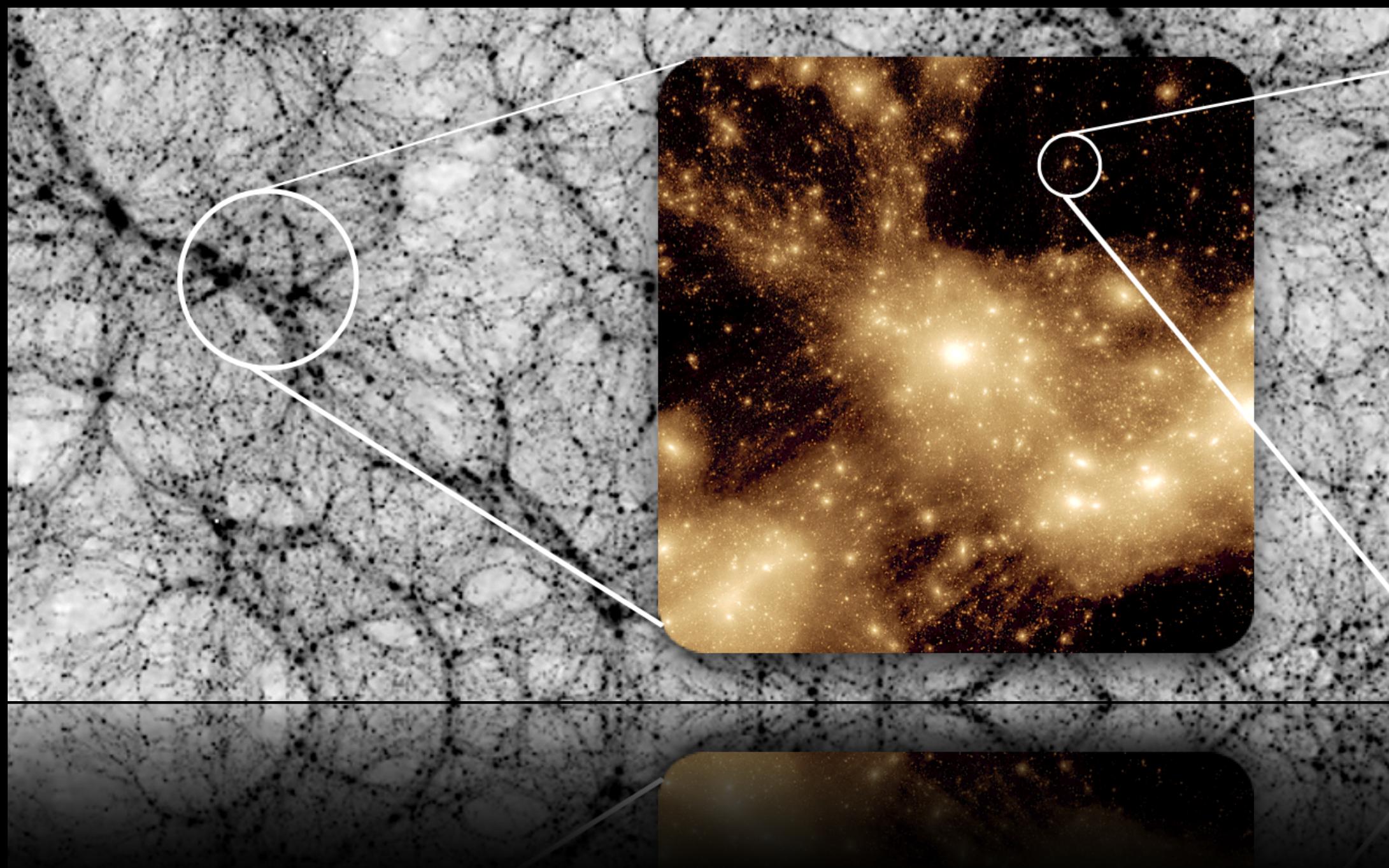
large-scale structure  
[ $\sim 10^9$  light-years]



# structures are coupled across multiple scales

large-scale structure  
[ $\sim 10^9$  light-years]

dark matter haloes  
[ $\sim 10^6$  light-years]

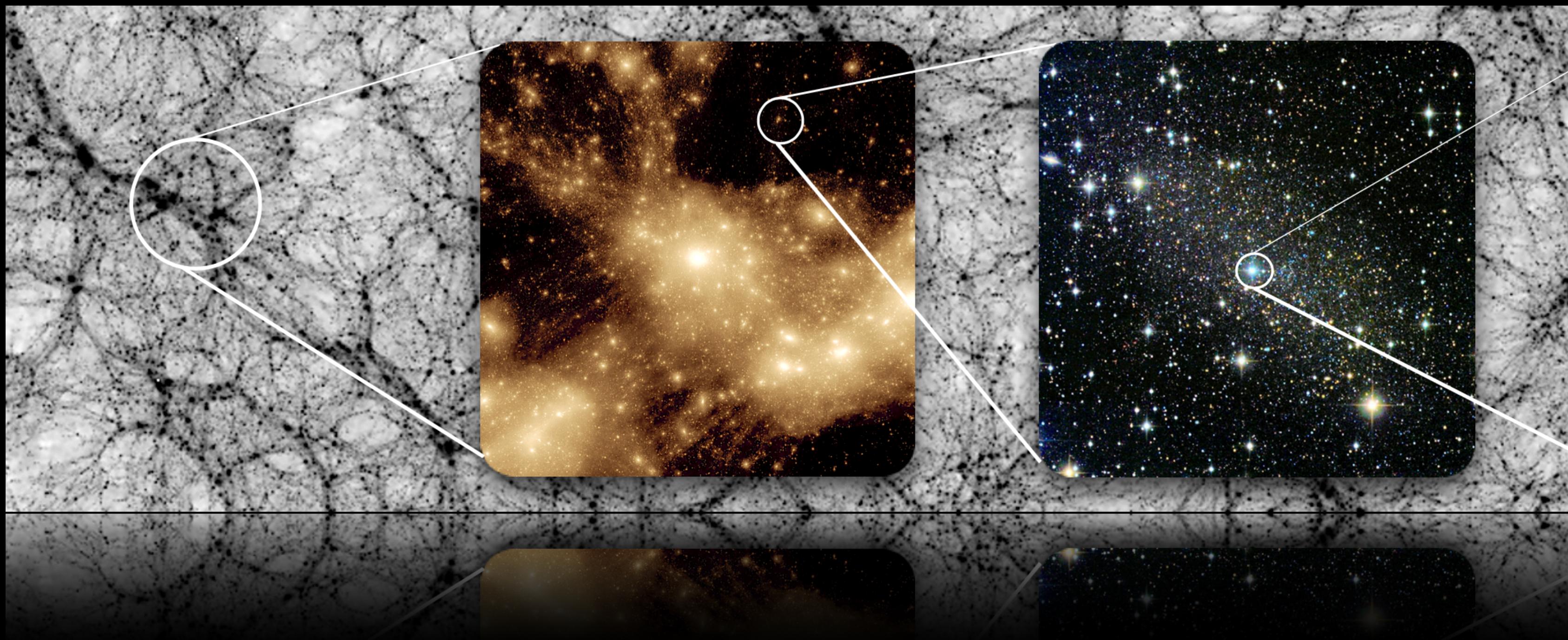


# structures are coupled across multiple scales

large-scale structure  
[ $\sim 10^9$  light-years]

dark matter haloes  
[ $\sim 10^6$  light-years]

galaxies  
[ $\sim 10^3$  light-years]



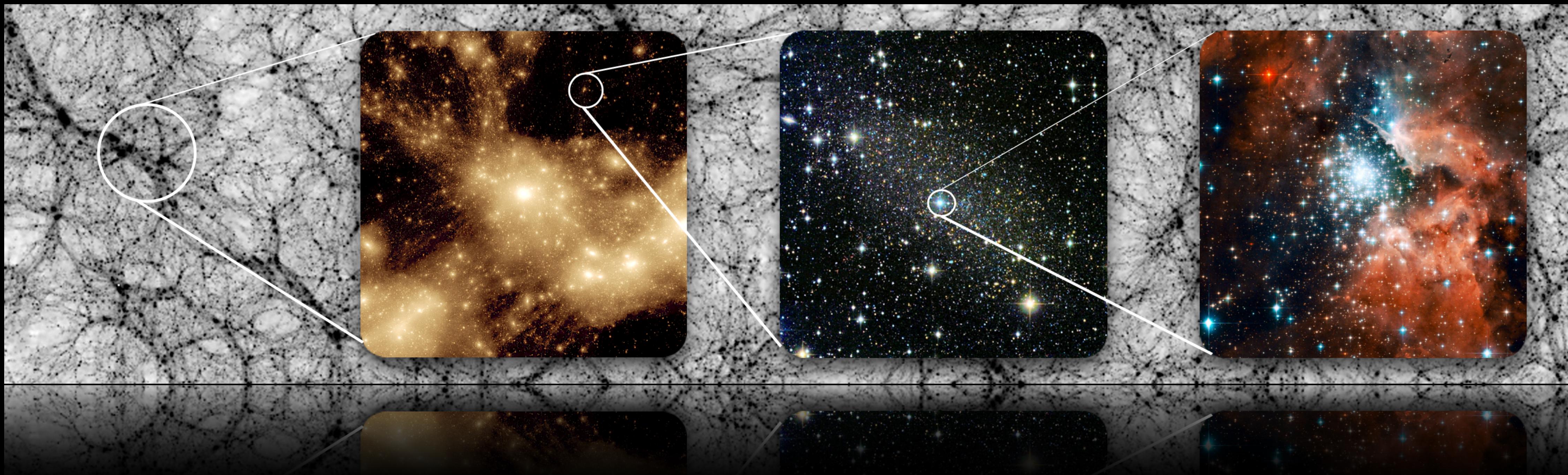
# structures are coupled across multiple scales

large-scale structure  
[ $\sim 10^9$  light-years]

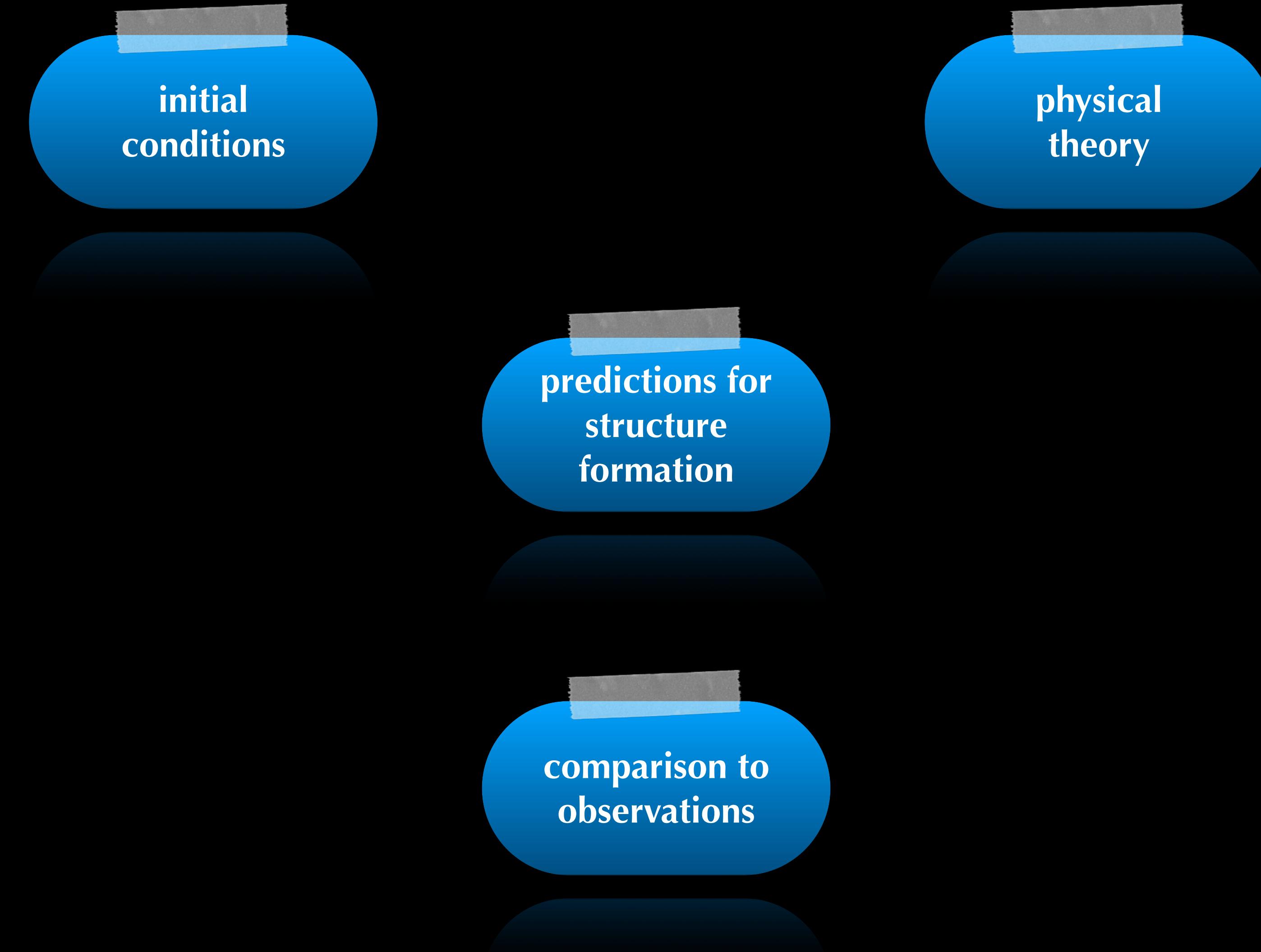
dark matter haloes  
[ $\sim 10^6$  light-years]

galaxies  
[ $\sim 10^3$  light-years]

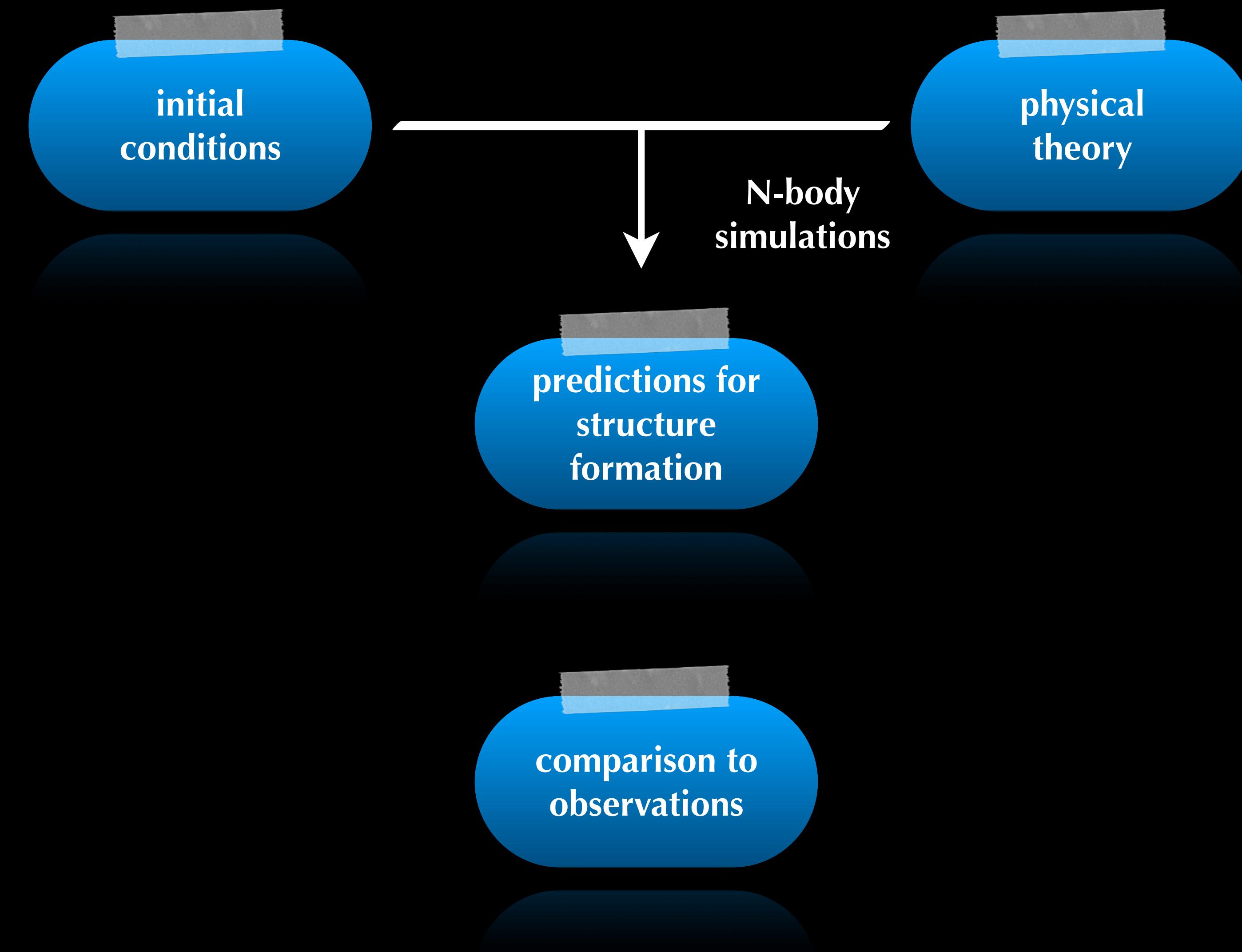
star formation sites  
[ $\sim$  light-years]



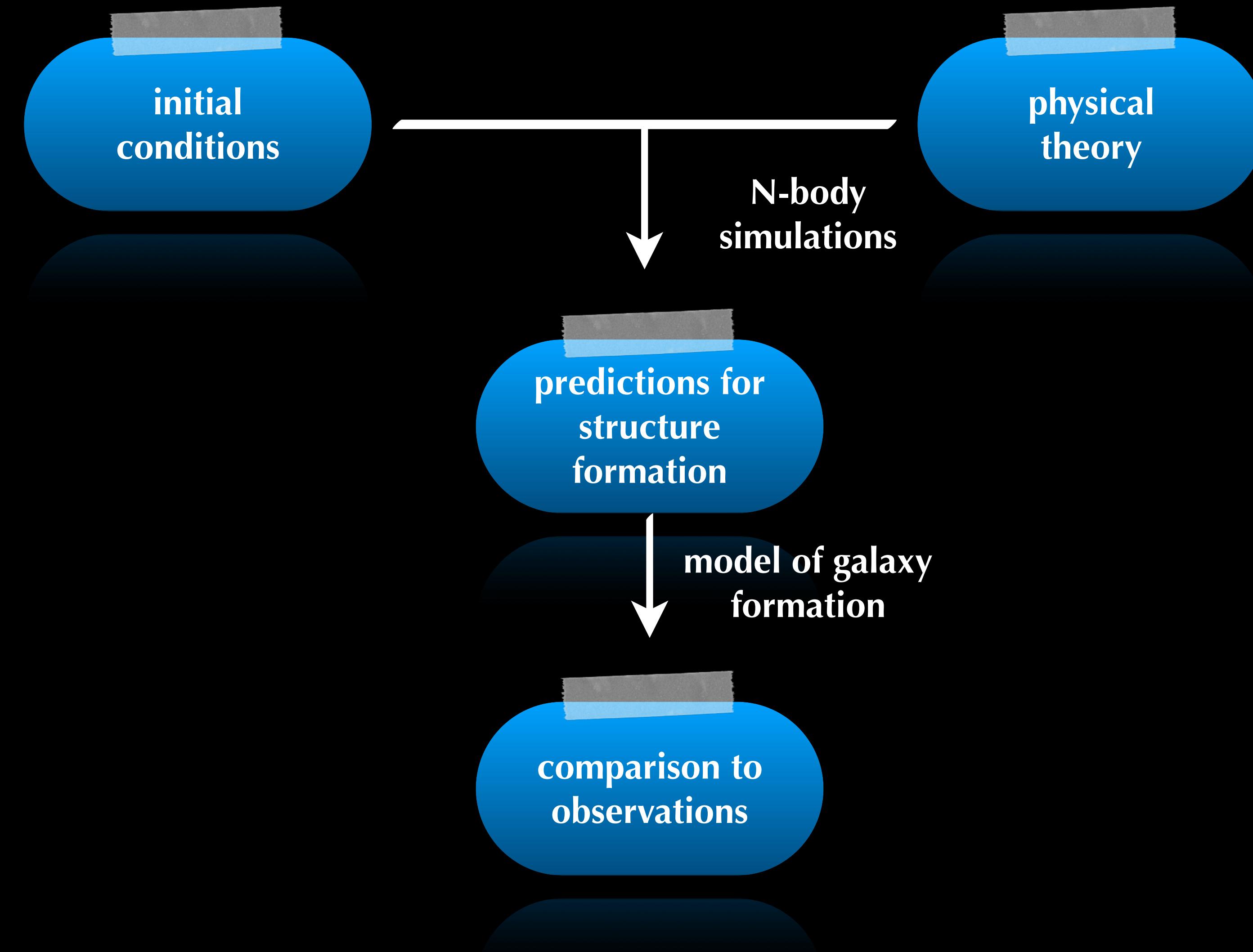
# the pathway to simulating the cosmos



# the pathway to simulating the cosmos



# the pathway to simulating the cosmos

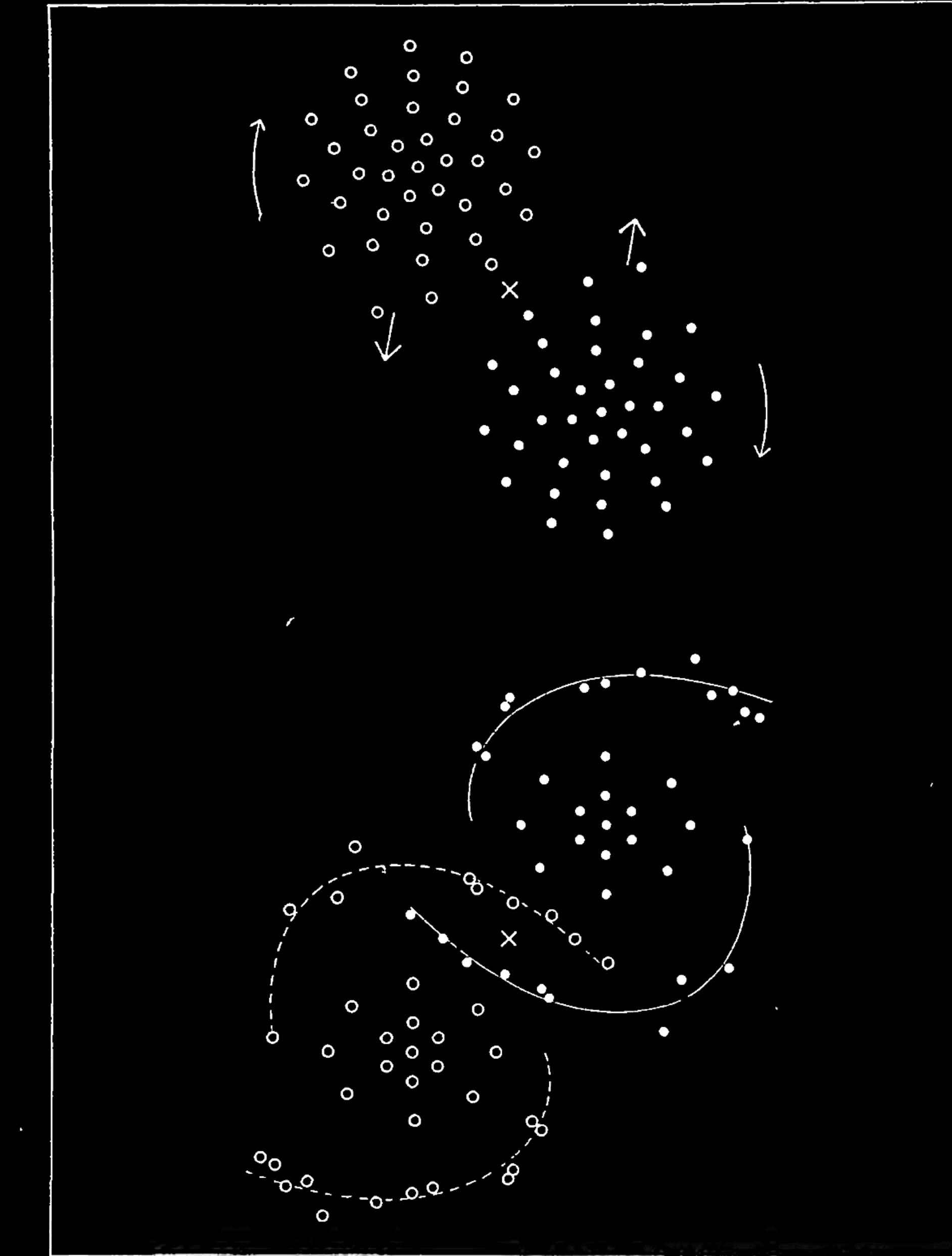


a little bit of history

Erik Holmberg



Holmberg (1941)  
the tidal encounter of two  
“extragalactic nebulae”



**White (1977);  
700 particles**

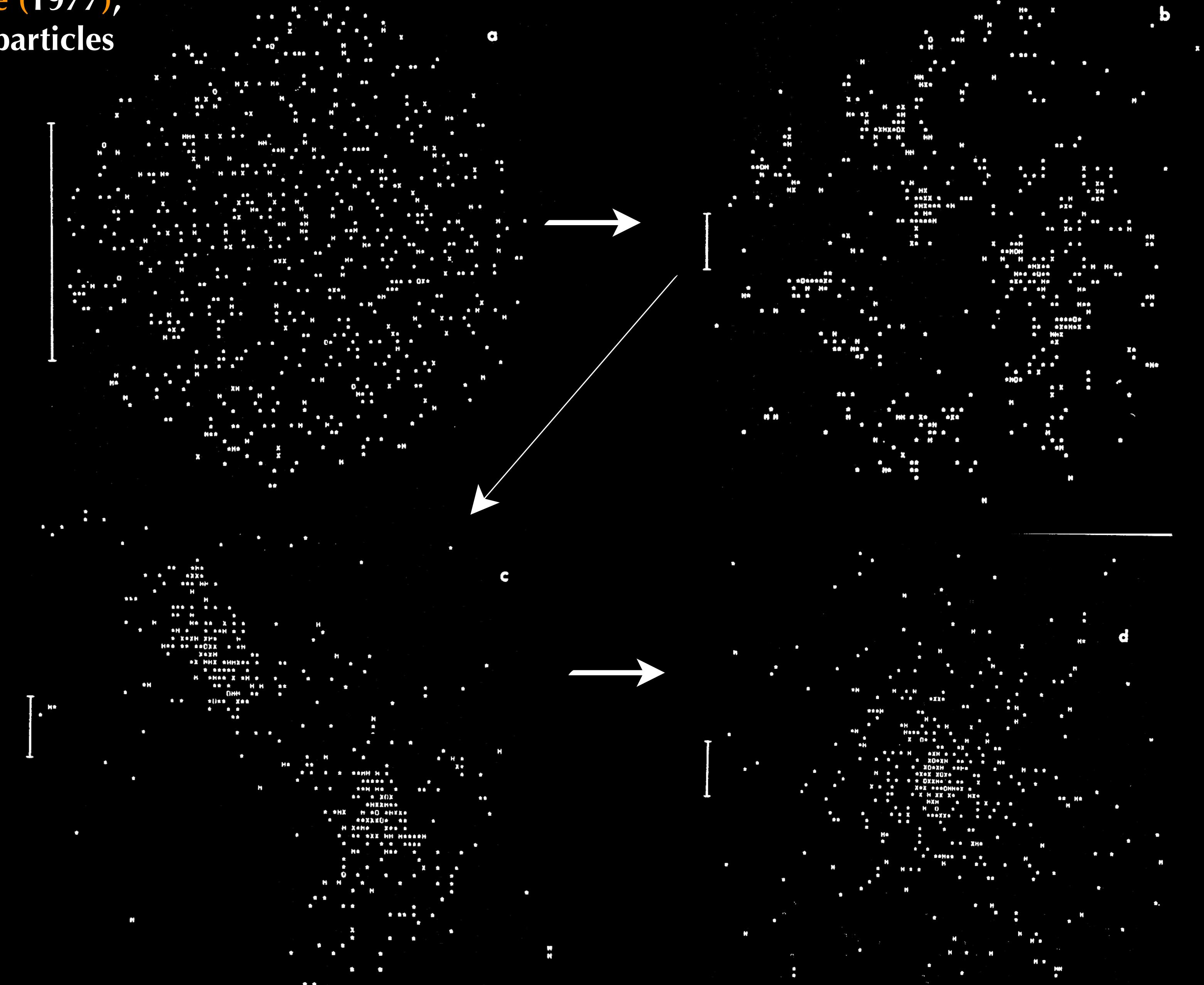


FIG. 1(c,d)

**White (1977);  
700 particles**

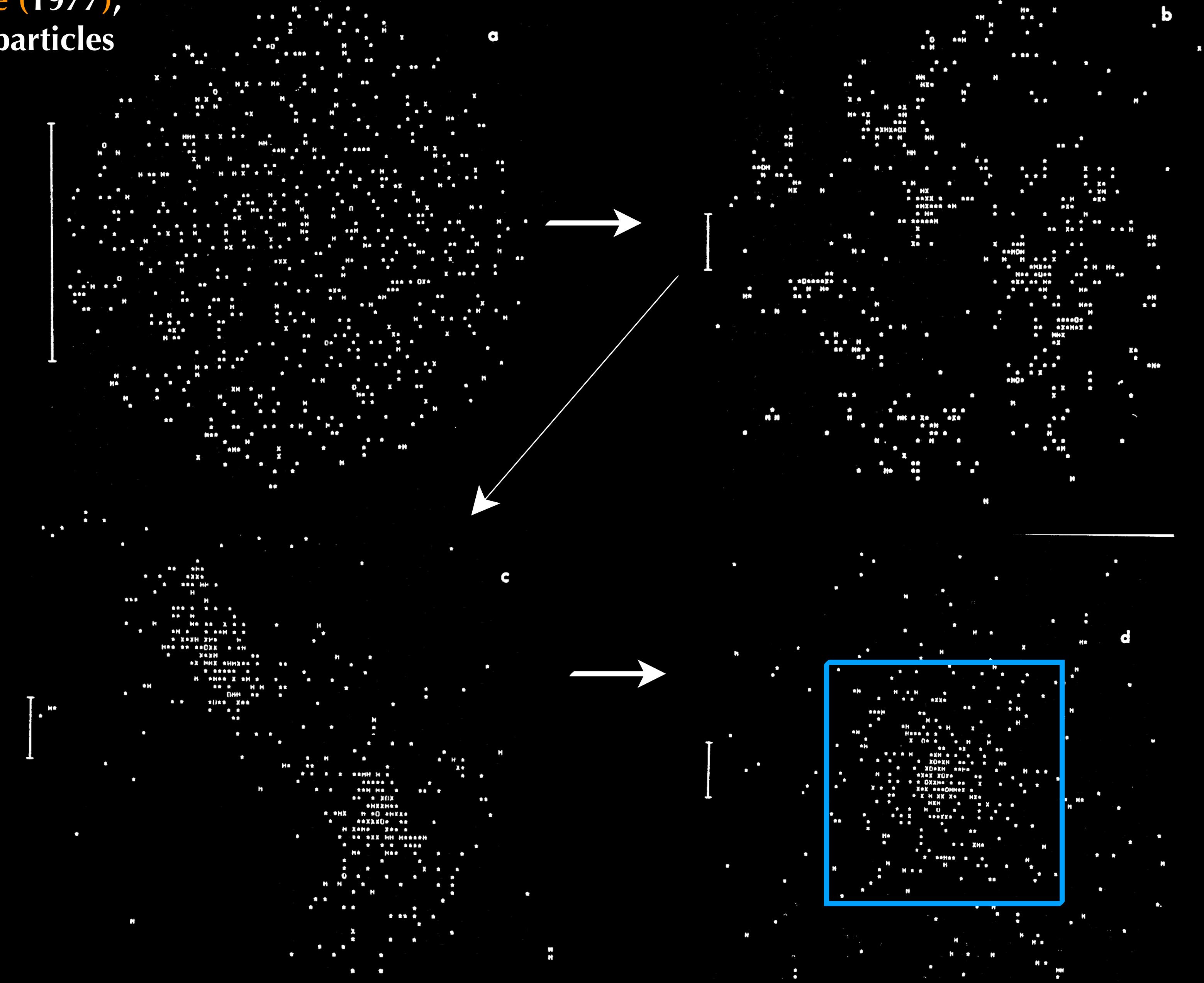


FIG. 1(c,d)

d

# the rise of particle dark matter

Volume 94B, number 2

PHYSICS LETTERS

28 July 1980

## AN ESTIMATE OF THE $\nu_e$ MASS FROM THE $\beta$ -SPECTRUM OF TRITIUM IN THE VALINE MOLECULE

V.A. LUBIMOV, E.G. NOVIKOV, V.Z. NOZIK, E.F. TRETYAKOV and V.S. KOSIK<sup>1</sup>

*Institute of Theoretical and Experimental Physics, Moscow, USSR*

Received 4 June 1980

$$14 \leq M_\nu \leq 46 \text{ eV} \quad (99\% \text{ C.L.}) .$$

The high energy part of the  $\beta$ -spectrum was measured by a toroidal  $\beta$ -spectrometer. The resu

We consider this as an indication that the electron antineutrino has a non-zero mass. For the time being we do not see any effects which could have shifted essentially the above-mentioned limits. We continue the experimental study of the  $\beta$ -spectrum of tritium.

# early candidates: the neutrino (~1980s)

weak interactions



not visible

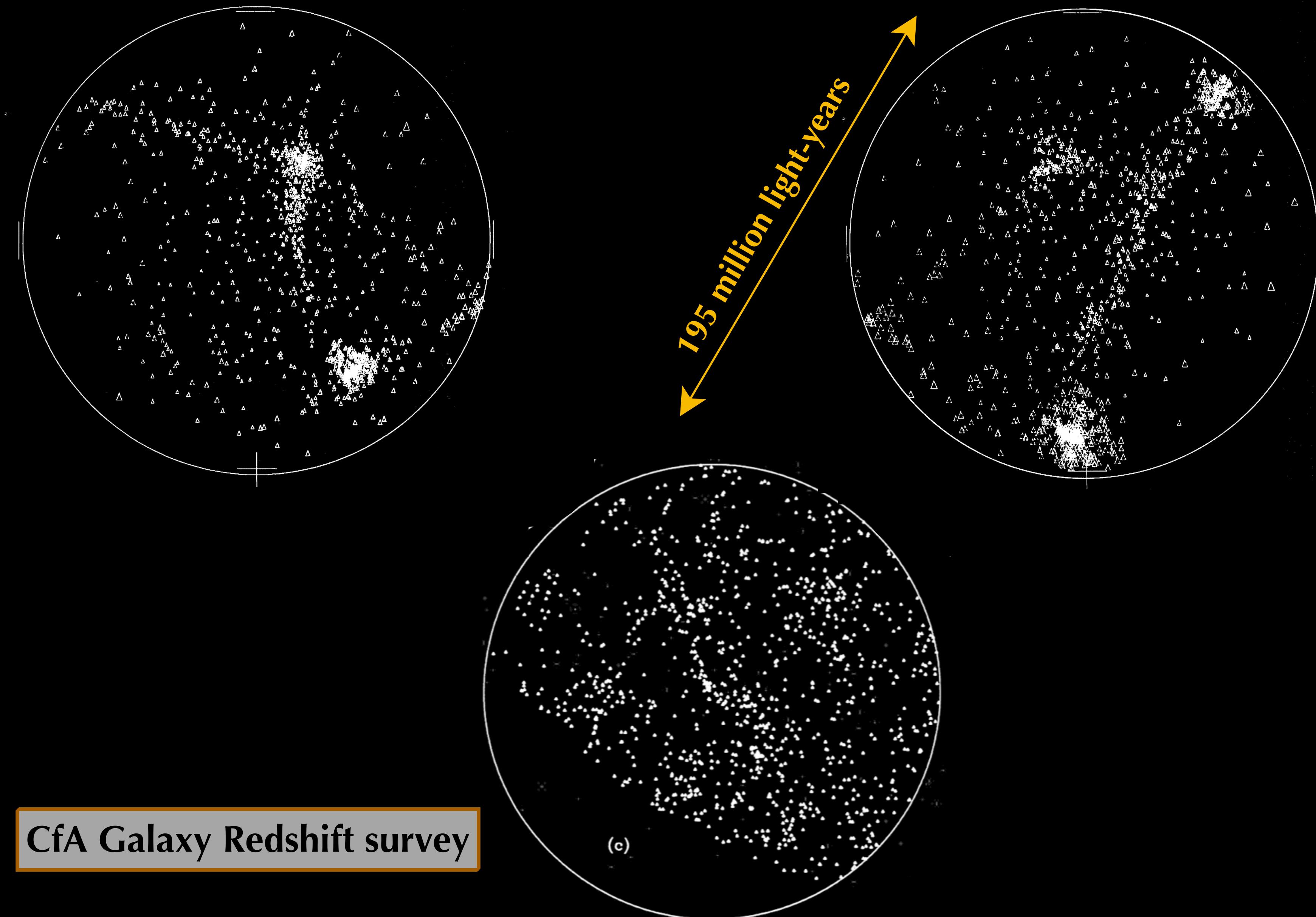


known particle



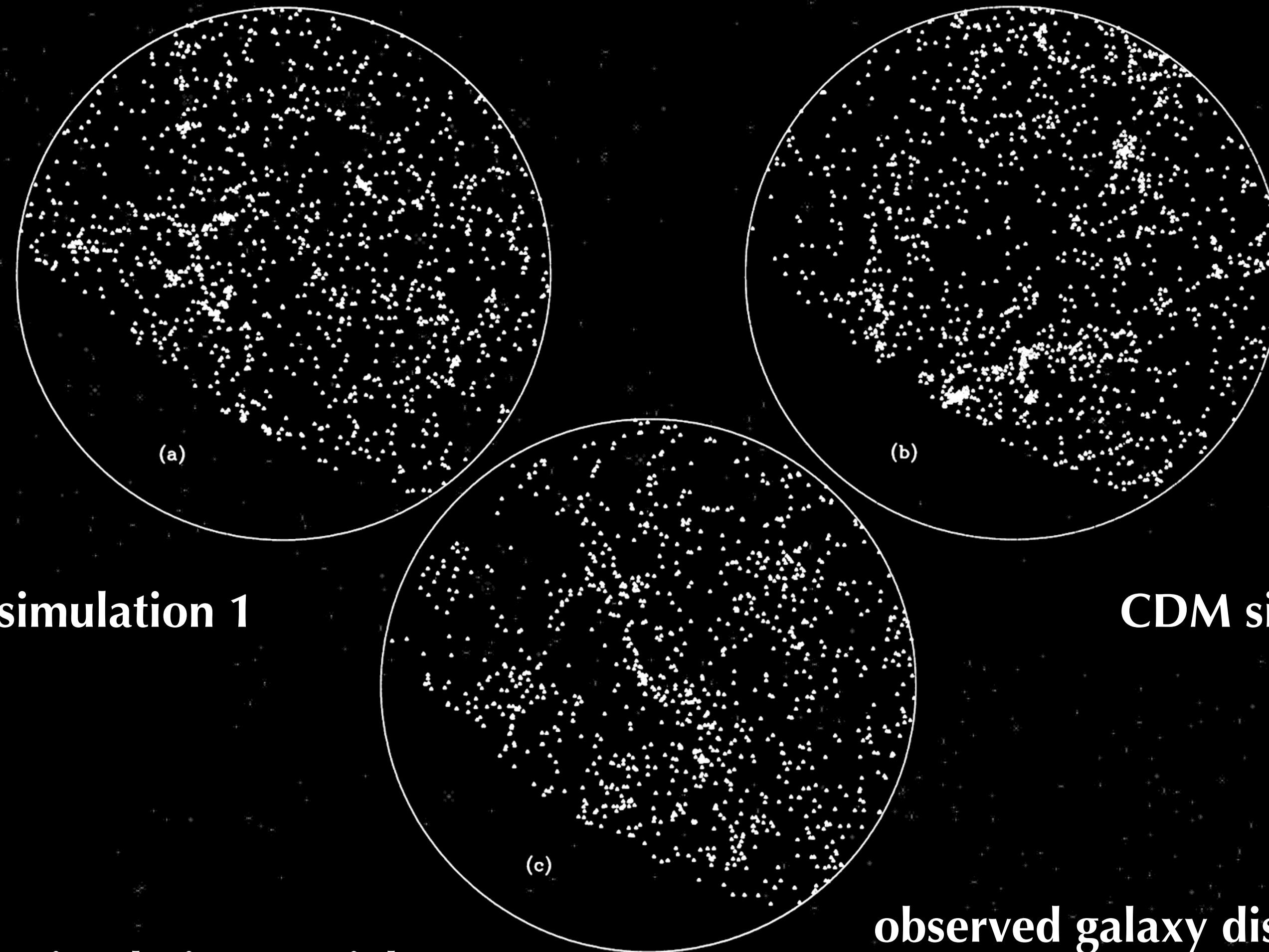
neutrinos are (almost) massless — zip around the Universe near the speed of light

hot dark matter



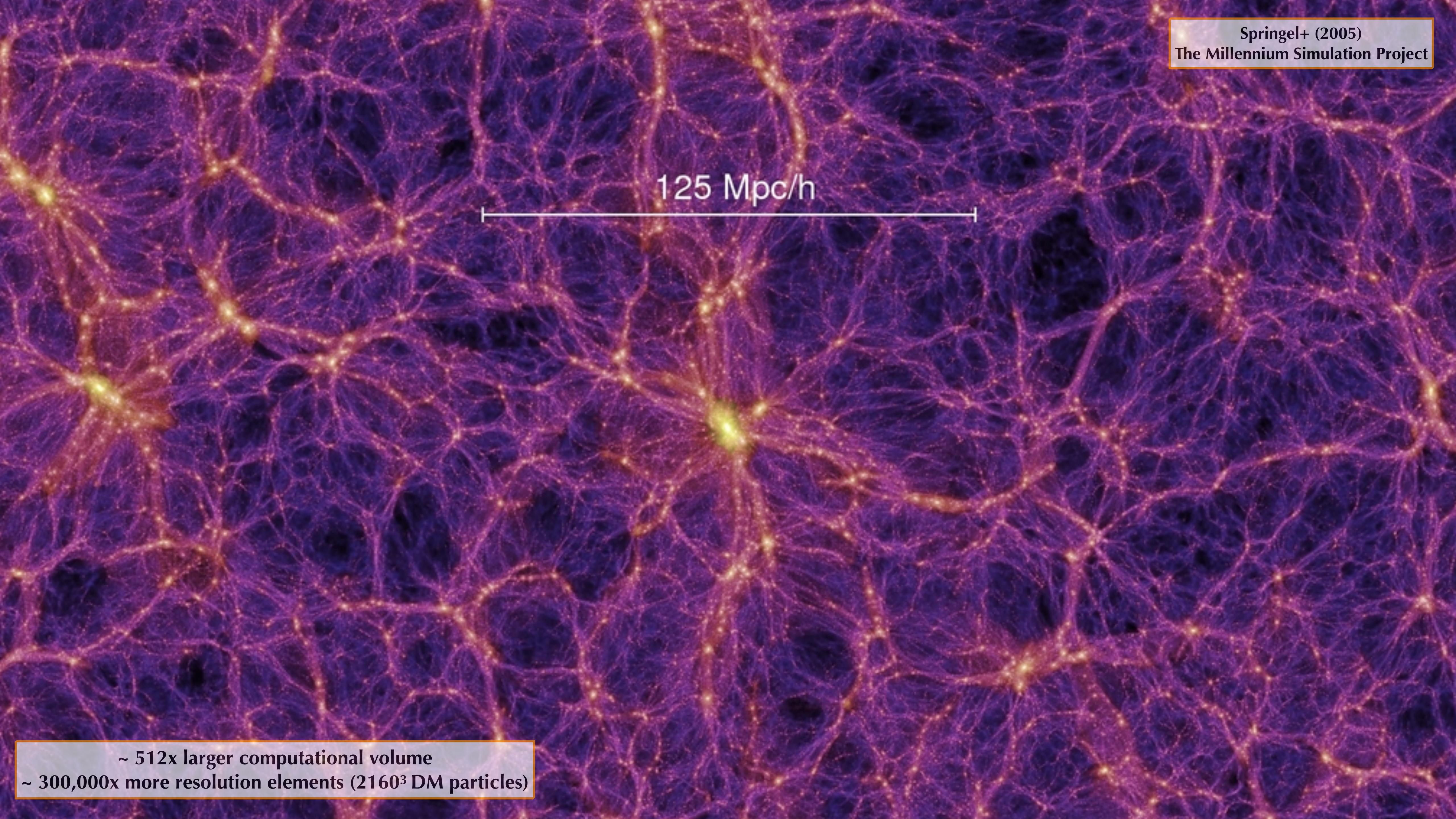
Klypin & Shandarin (1983);  $32^3$  simulation particles

# the emergence of **cold** dark matter



Davis+ (1985);  $32^3$  simulation particles

observed galaxy distribution

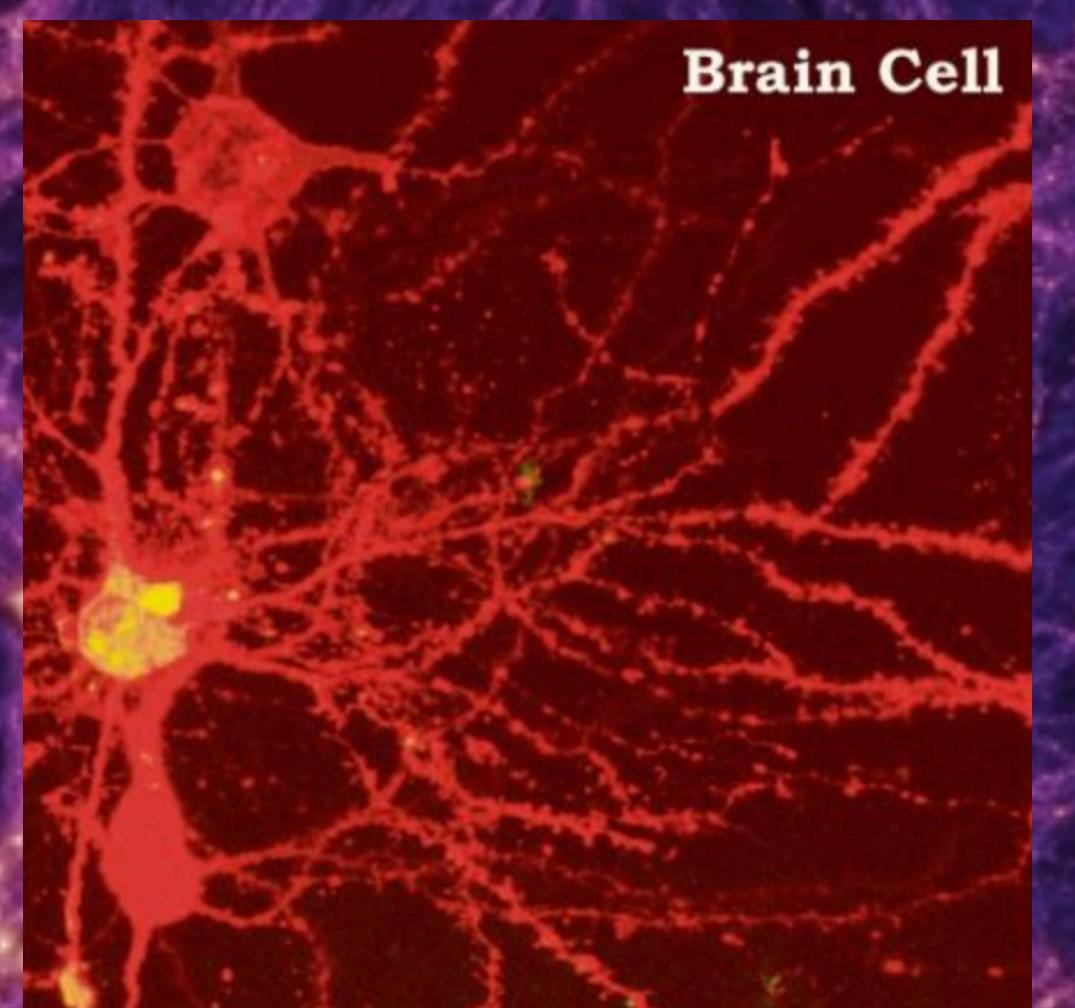


125 Mpc/h

~ 512x larger computational volume

~ 300,000x more resolution elements ( $2160^3$  DM particles)

125 Mpc/h



~ 512x larger computational volume  
~ 300,000x more resolution elements ( $2160^3$  DM particles)

$z = 48.4$

$T = 0.05 \text{ Gyr}$



$z = 48.4$

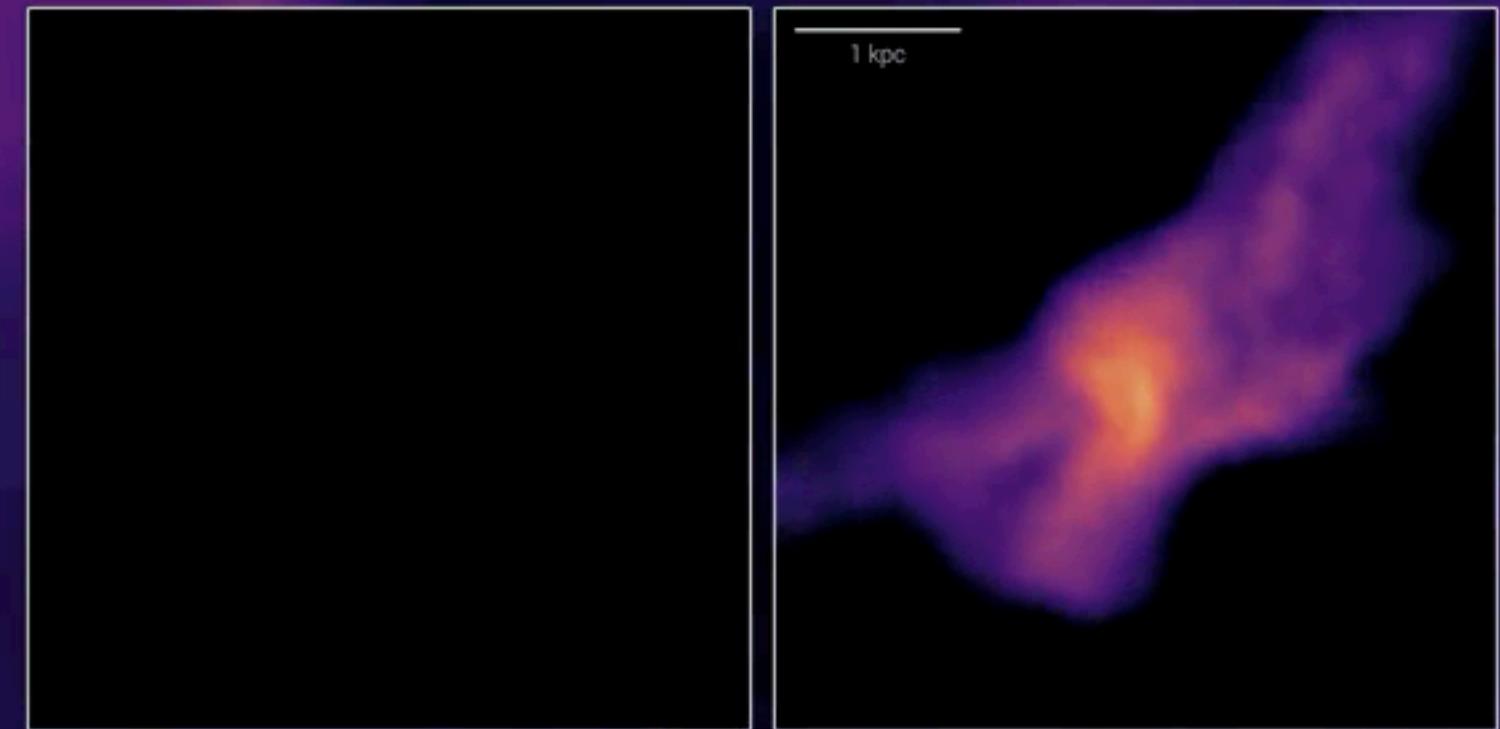
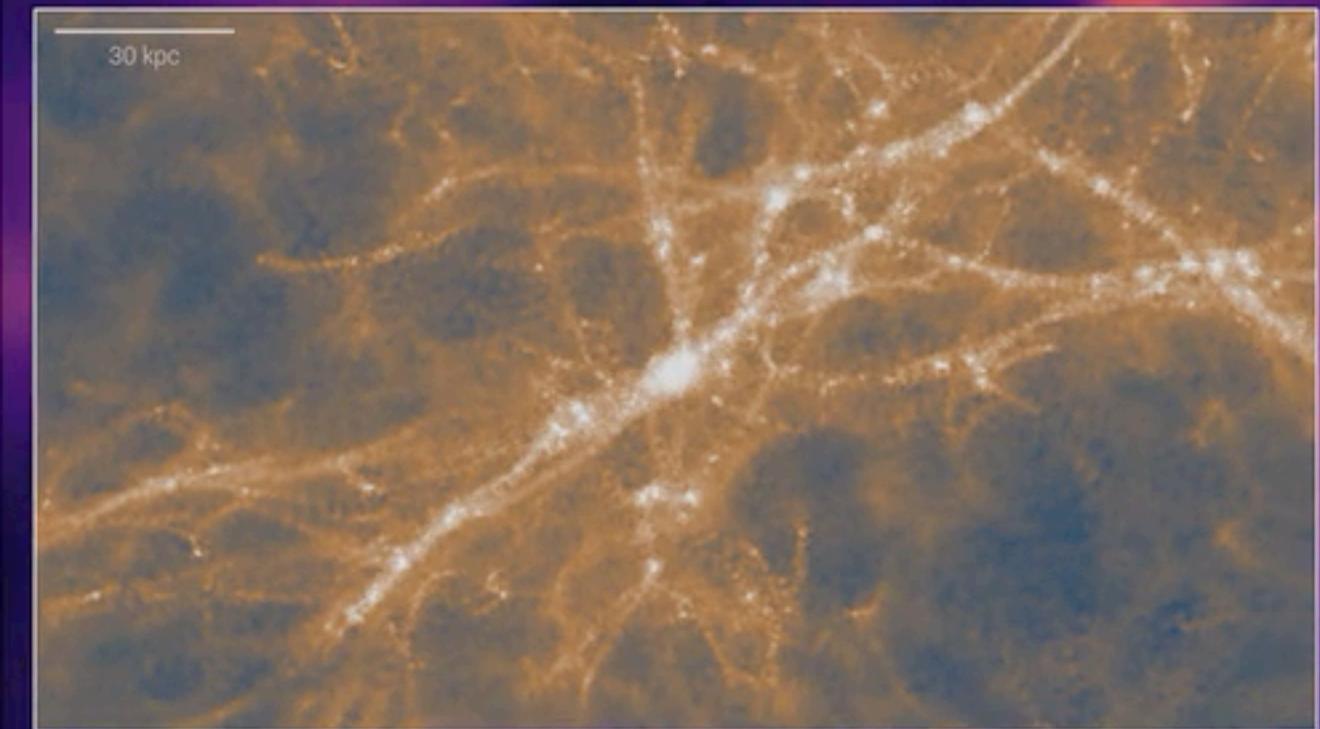
$T = 0.05 \text{ Gyr}$



9 kpc

$z = 9.8$

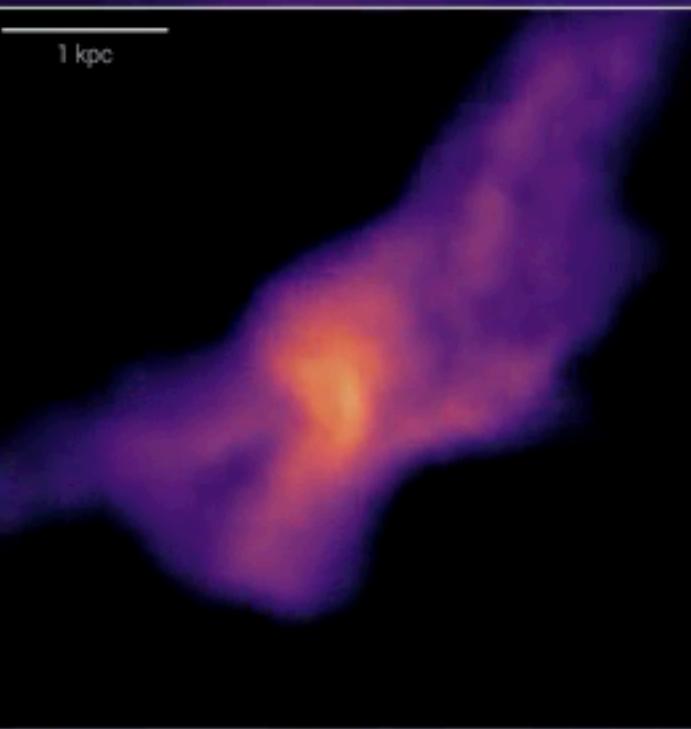
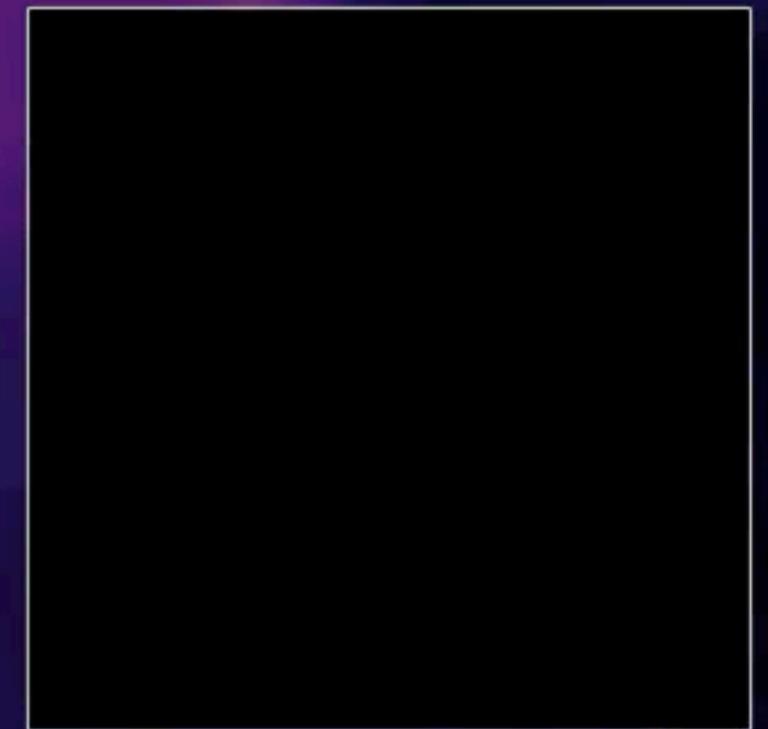
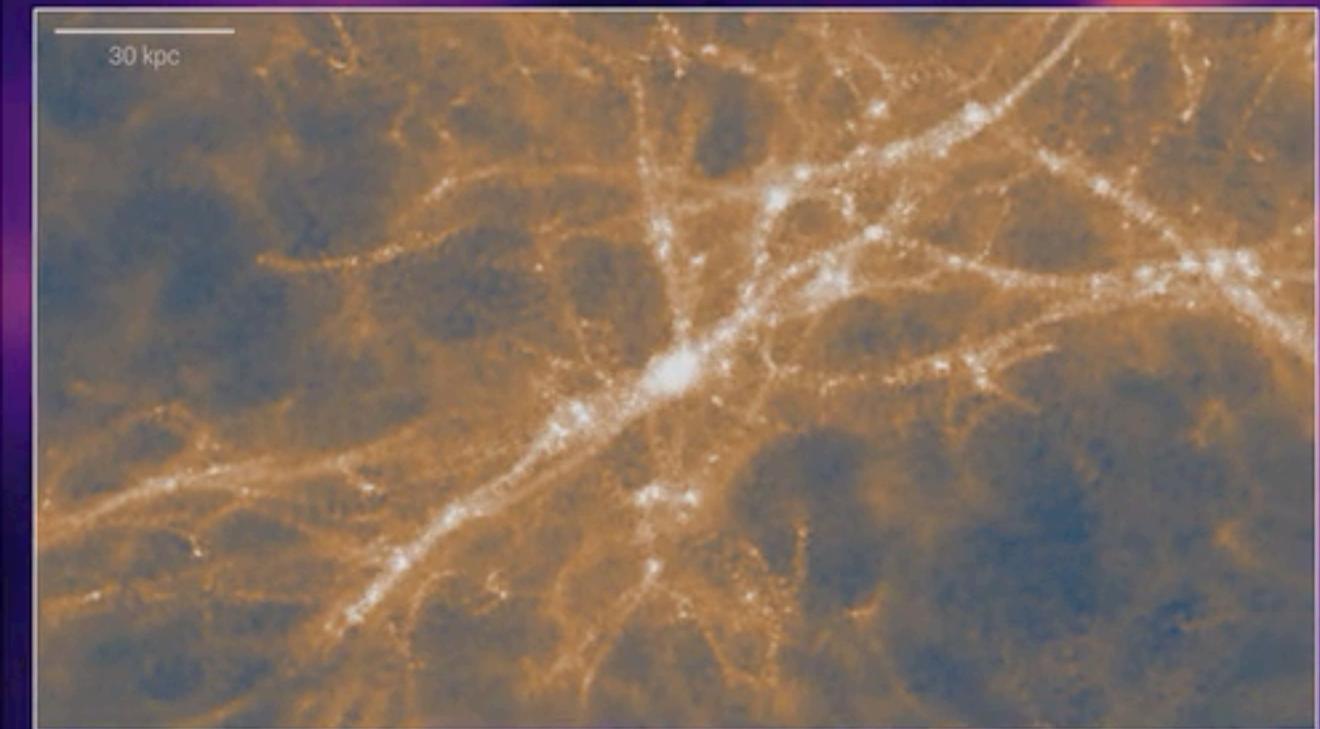
$\log M_\star = 7.74$   
 $SFR = 0.2 M_\odot \text{ yr}^{-1}$

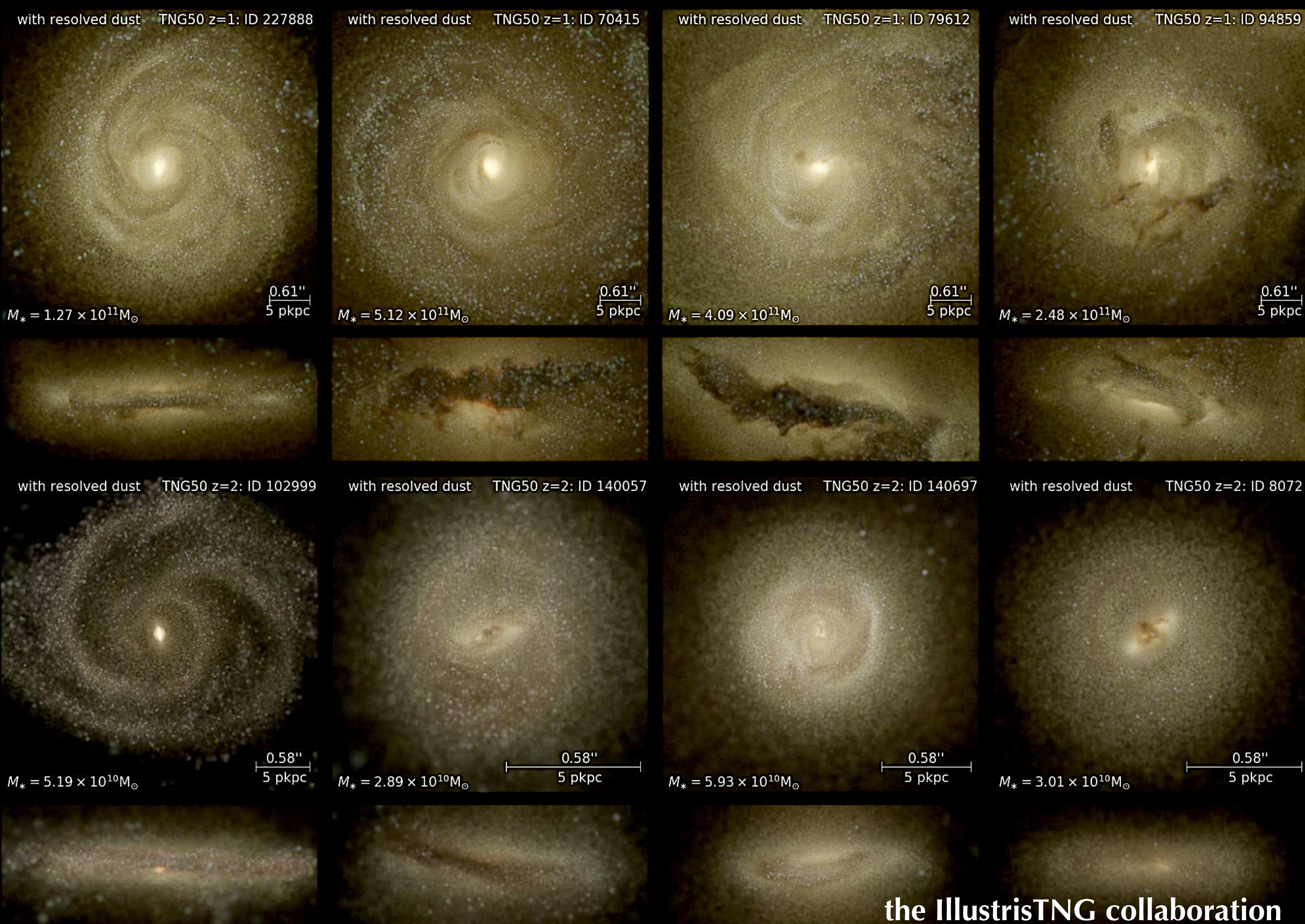


9 kpc

$z = 9.8$

$\log M_\star = 7.74$   
 $SFR = 0.2 M_\odot \text{ yr}^{-1}$

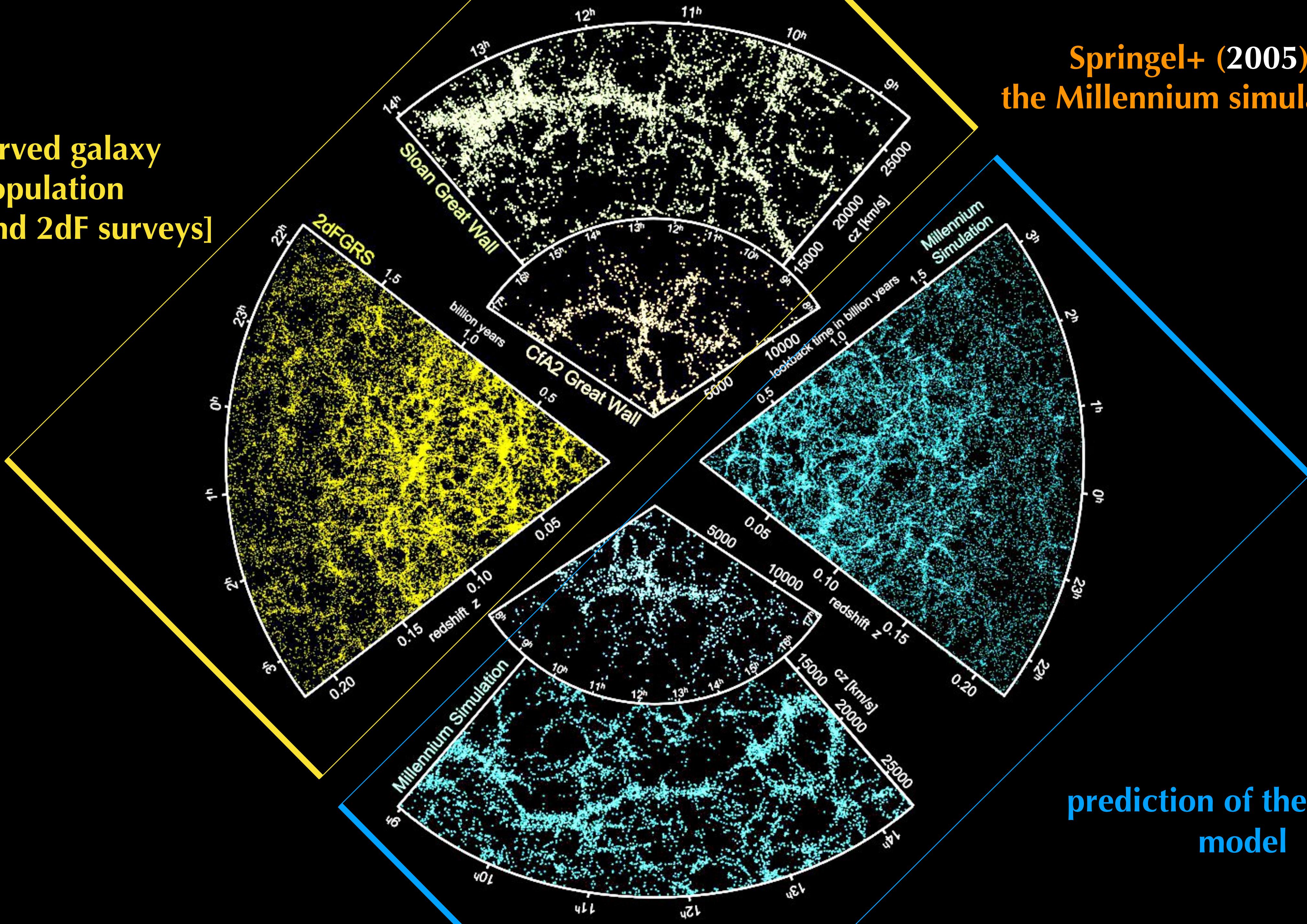




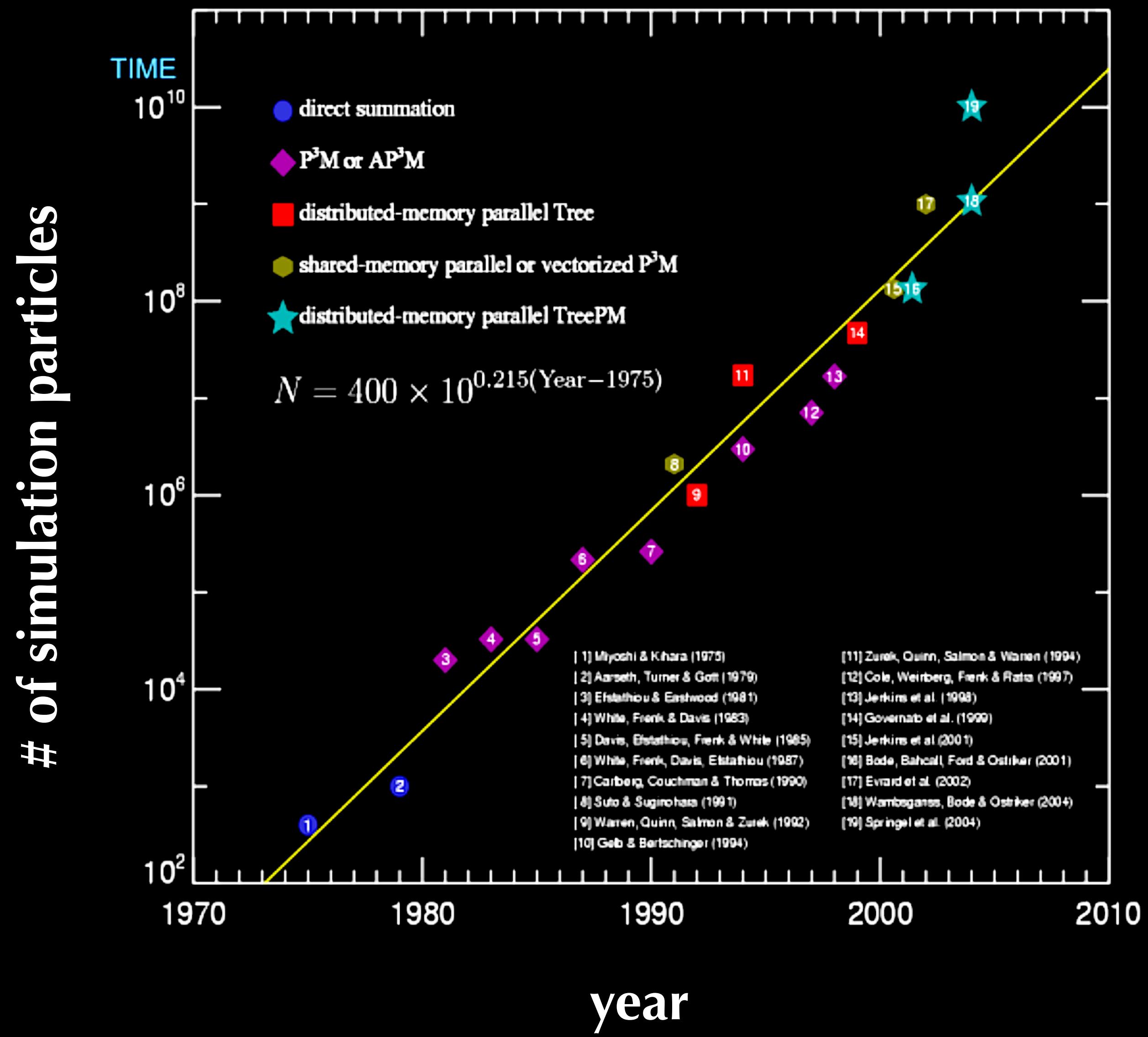
the IllustrisTNG collaboration

Springel+ (2005)  
the Millennium simulation

observed galaxy  
population  
[SDSS and 2dF surveys]

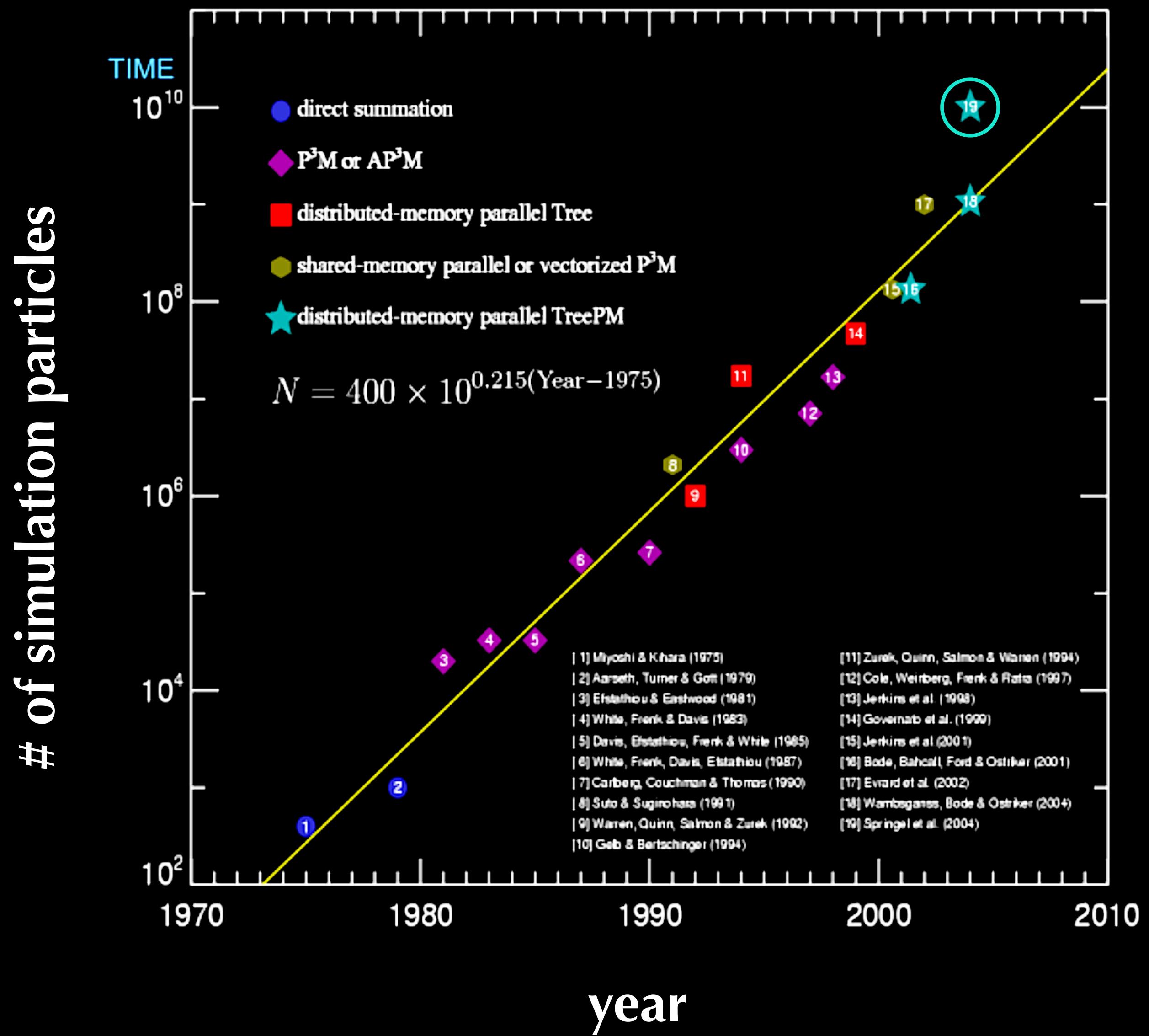


prediction of the  $\Lambda$ CDM  
model



**Moore's Law —**  
computers double their  
speed every ~18 months  
or so

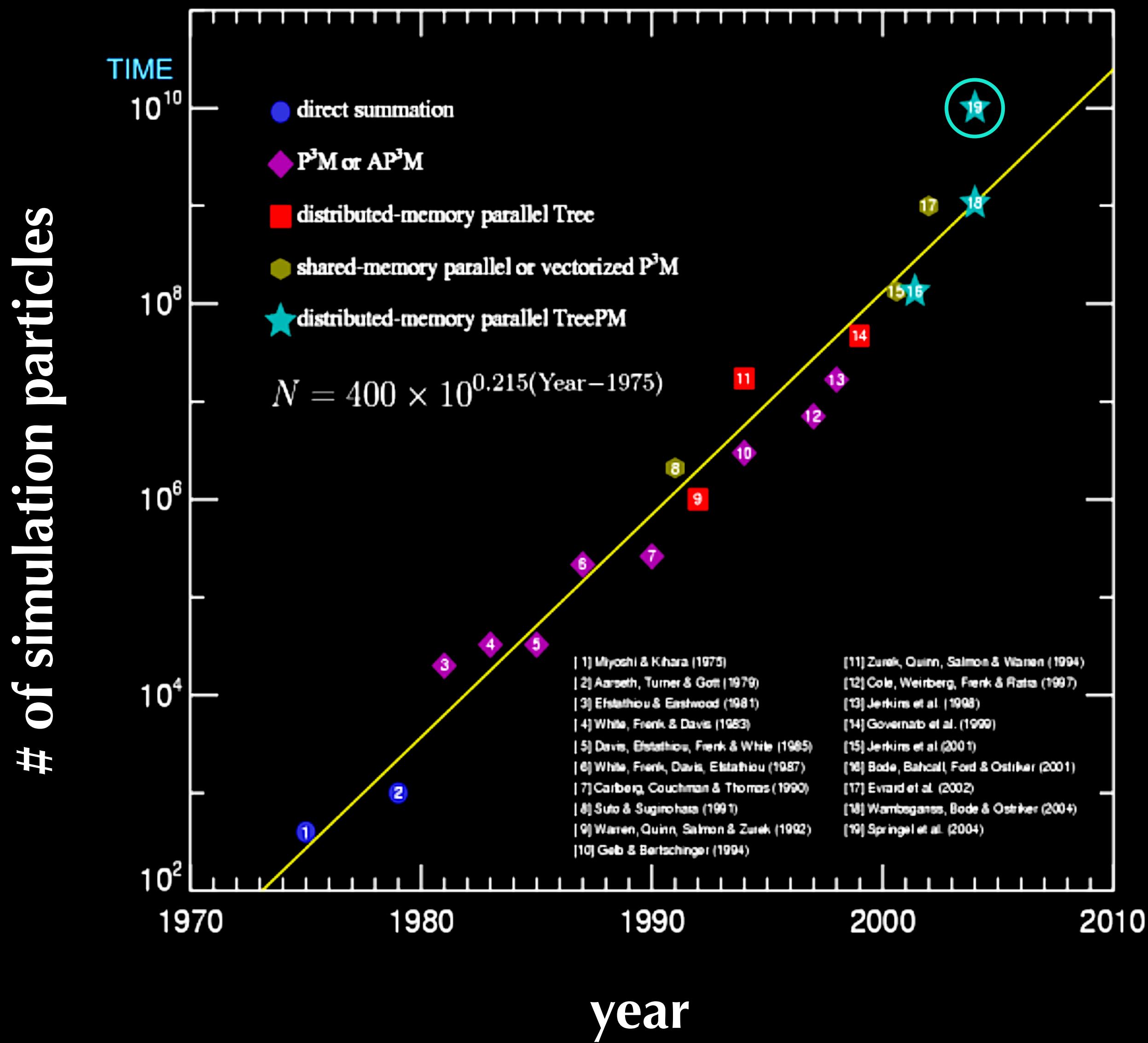
N-body simulations have  
roughly doubled in size  
every 16-17 months



**Moore's Law —**  
computers double their  
speed every ~18 months  
or so

N-body simulations have  
roughly doubled in size  
every 16-17 months

the Millennium simulation is  
a bit of an outlier — should  
have “only been possible” in  
2010 — but performed in  
2004/5!



**Moore's Law —**  
computers double their  
speed every ~18 months  
or so

N-body simulations have  
roughly doubled in size  
every 16-17 months

the Millennium simulation is  
a bit of an outlier — should  
have “only been possible” in  
2010 — but performed in  
2004/5!

the **N-body** method

# Poisson-Vlasov equation

## [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

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

# Poisson-Vlasov equation

## [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

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

collisionless Boltzmann  
equation

# Poisson-Vlasov equation

## [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

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

collisionless Boltzmann  
equation

the “**distribution function**”  
[number density of particles at  
position  $\mathbf{x}$  with velocity  $\mathbf{v}$ ]

## Poisson-Vlasov equation [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$


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

collisionless Boltzmann  
equation

the “**distribution function**”  
[number density of particles at  
position  $\mathbf{x}$  with velocity  $\mathbf{v}$ ]

## macroscopic DM [particles]

$$\ddot{\mathbf{x}}_i = - \nabla_i \Phi (\mathbf{x}_i)$$

$$\Phi (\mathbf{x}) = - G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

sample the phase-space of dark  
matter with macroscopic  
particles

## Poisson-Vlasov equation [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

collisionless Boltzmann  
equation

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

the “**distribution function**”  
[number density of particles at  
position  $\mathbf{x}$  with velocity  $\mathbf{v}$ ]

## macroscopic DM [particles]

$$\ddot{\mathbf{x}}_i = - \nabla_i \Phi(\mathbf{x}_i)$$

softening

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

sample the phase-space of dark  
matter with macroscopic  
particles

## Poisson-Vlasov equation [fluid]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left( -\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

collisionless Boltzmann  
equation

the “**distribution function**”  
[number density of particles at  
position  $\mathbf{x}$  with velocity  $\mathbf{v}$ ]

## macroscopic DM [particles]

$$\ddot{\mathbf{x}}_i = - \nabla_i \Phi(\mathbf{x}_i)$$

**softening**

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

sample the phase-space of dark  
matter with macroscopic  
particles

→ need large  $N$  to be valid!

# relaxation time in an N-body system

*N*-body systems undergo gravitational encounters or “collisions” in a process that is referred to as **relaxation**. the rate at which this happens is known as the **relaxation time**:

# relaxation time in an N-body system

*N*-body systems undergo gravitational encounters or “collisions” in a process that is referred to as **relaxation**. the rate at which this happens is known as the **relaxation time**:

$$t_{\text{relax}} \simeq \frac{N}{8 \ln(R/\epsilon)} t_{\text{cross}}$$

$$t_{\text{cross}} \simeq \frac{R}{\sigma_v}$$

# relaxation time in an N-body system

*N*-body systems undergo gravitational encounters or “collisions” in a process that is referred to as **relaxation**. the rate at which this happens is known as the **relaxation time**:

$$t_{\text{relax}} \simeq \frac{N}{8 \ln (R/\epsilon)} t_{\text{cross}}$$

$$t_{\text{cross}} \simeq \frac{R}{\sigma_v}$$

characteristic  
size of system

velocity  
dispersion

# relaxation time in an N-body system

*N*-body systems undergo gravitational encounters or “collisions” in a process that is referred to as **relaxation**. the rate at which this happens is known as the **relaxation time**:

$$t_{\text{relax}} \simeq \frac{N}{8 \ln(R/\epsilon)} t_{\text{cross}}$$

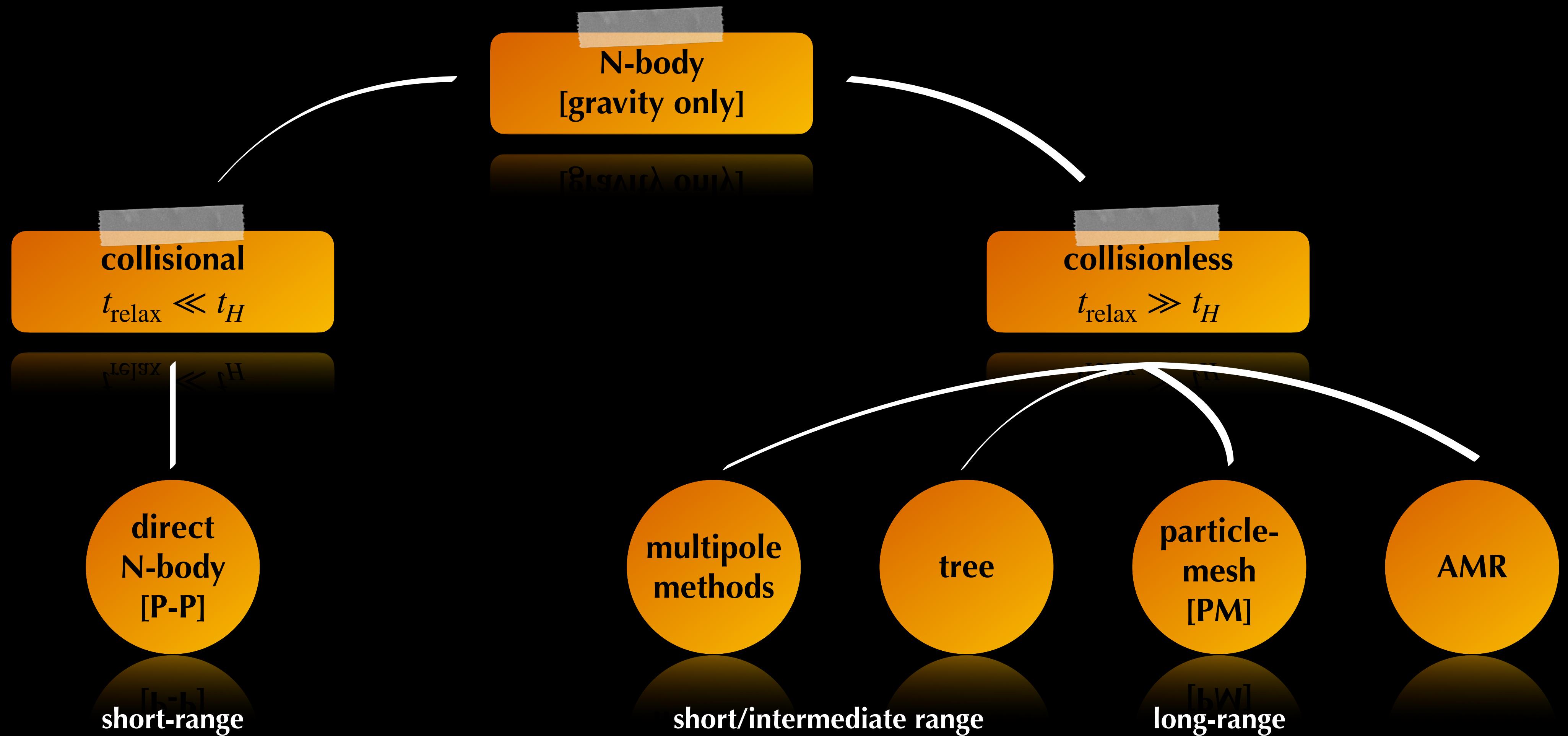
$$t_{\text{cross}} \simeq \frac{R}{\sigma_v}$$

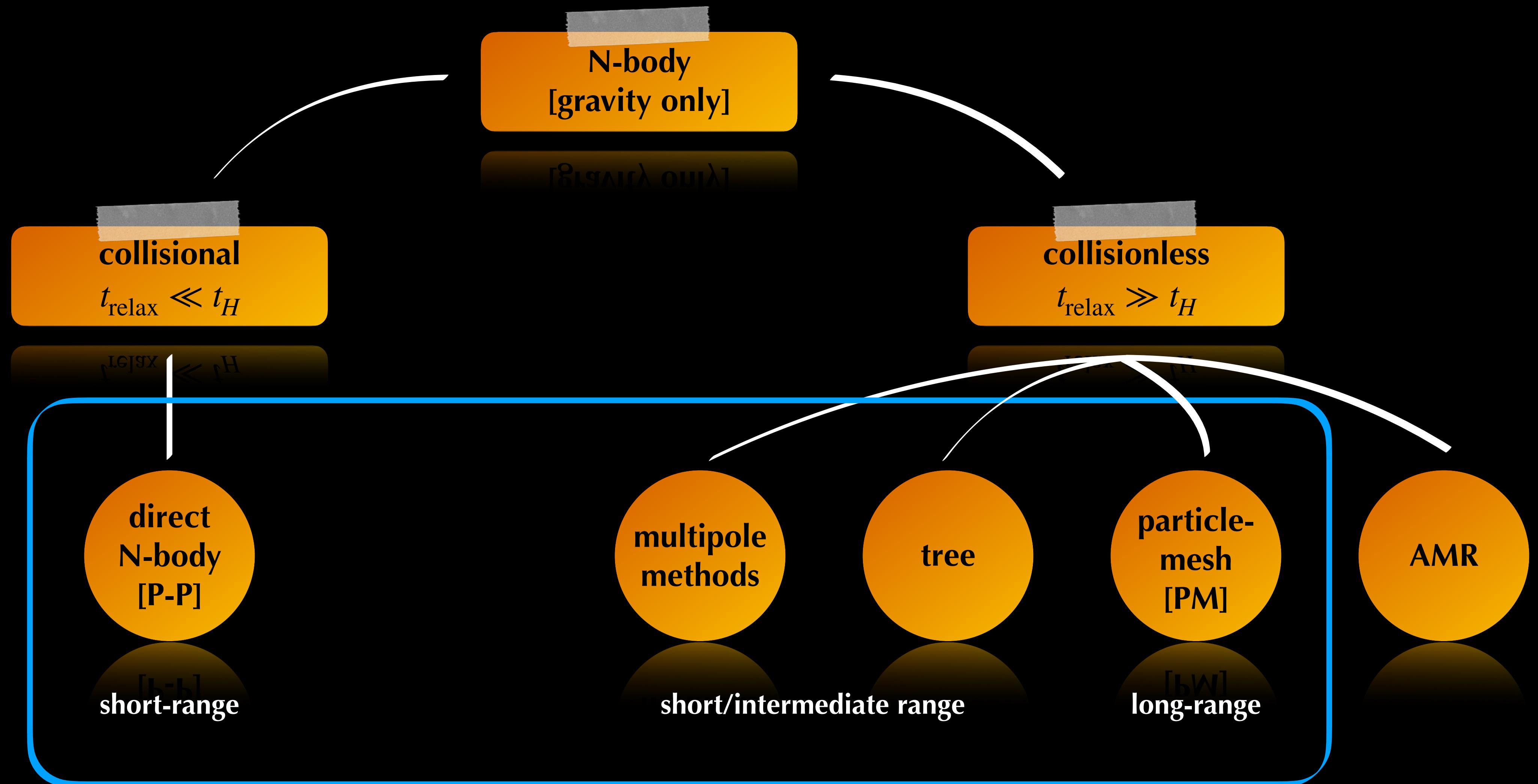
characteristic  
size of system

velocity  
dispersion

in a “real” dark matter halo,  $N \sim 10^{70}$ , so we can always treat it as a collisionless system. in a simulation, typically  $N \sim 10^9$ , so relaxation timescale is dramatically shorter

$t_{\text{relax}} \gg t_H$       collisionless





hybrid methods: GADGET, SWIFT, PKDgrav

# direct N-body: particle-particle method

$$\ddot{x}_i = - \nabla \Phi(x_i)$$

$$\Phi(x) = -G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

this is the simplest N-body algorithm one can cook up: simply compute the force on particle  $j$  due to all  $N - 1$  other particles, and repeat over all particles

# direct N-body: particle-particle method

$$\ddot{x}_i = - \nabla \Phi(x_i)$$

$$\Phi(x) = -G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

super-easy to understand  
general  
highly accurate

this is the simplest N-body algorithm one can cook up: simply compute the force on particle  $j$  due to all  $N - 1$  other particles, and repeat over all particles

# direct N-body: particle-particle method

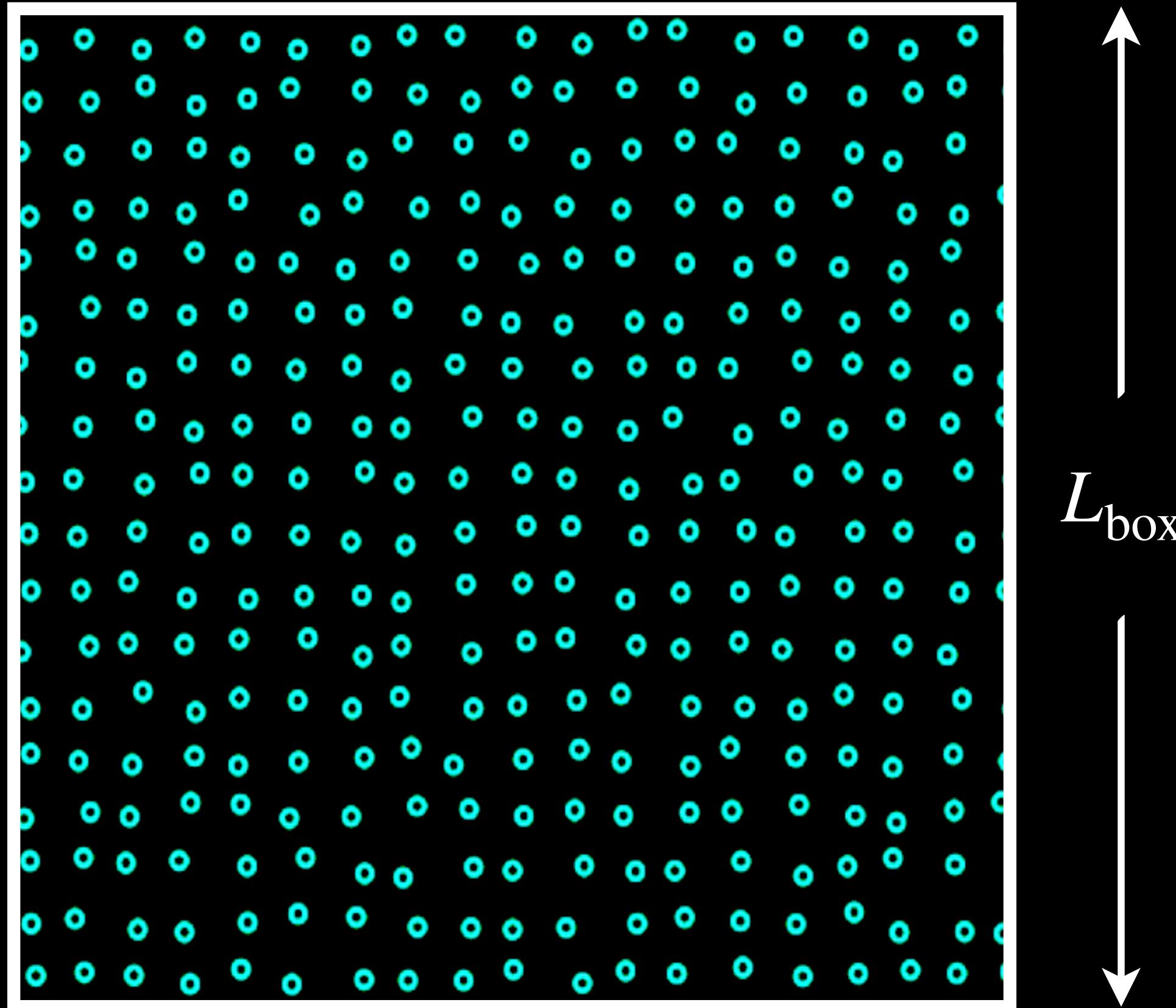
$$\ddot{x}_i = - \nabla \Phi(x_i)$$

$$\Phi(x) = -G \sum_{j=1}^N \frac{m_j}{\left[ (\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2 \right]^{1/2}}$$

super-easy to understand  
general  
highly accurate  
**computationally extremely expensive**  
 $\mathcal{O}(N^2)$

this is the simplest N-body algorithm one can cook up: simply compute the force on particle  $j$  due to all  $N - 1$  other particles, and repeat over all particles

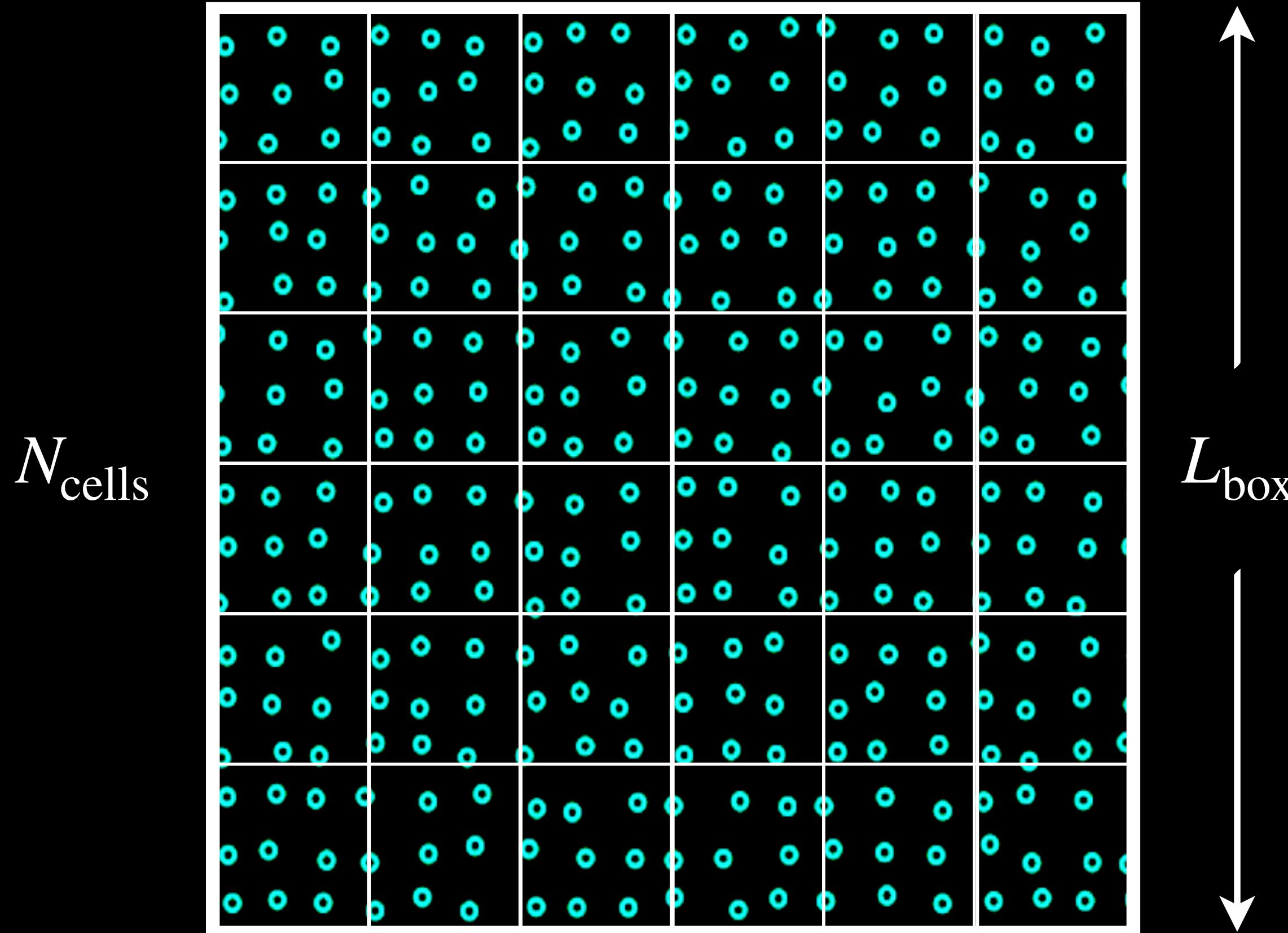
# particle-mesh method



use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# particle-mesh method

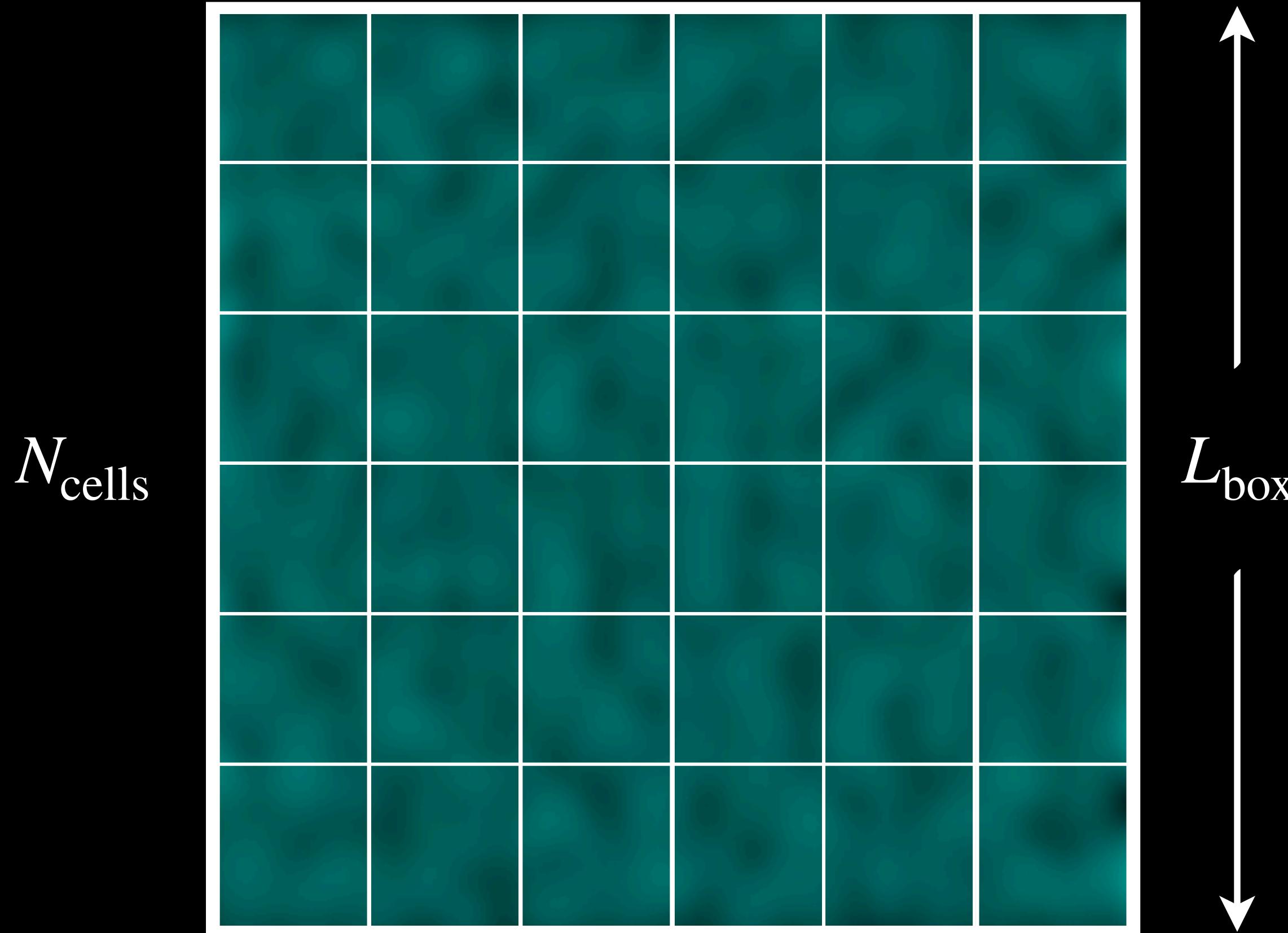


1. divide computational volume into a grid of size  $N_{\text{cells}} \times N_{\text{cells}} \times N_{\text{cells}}$

use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# particle-mesh method

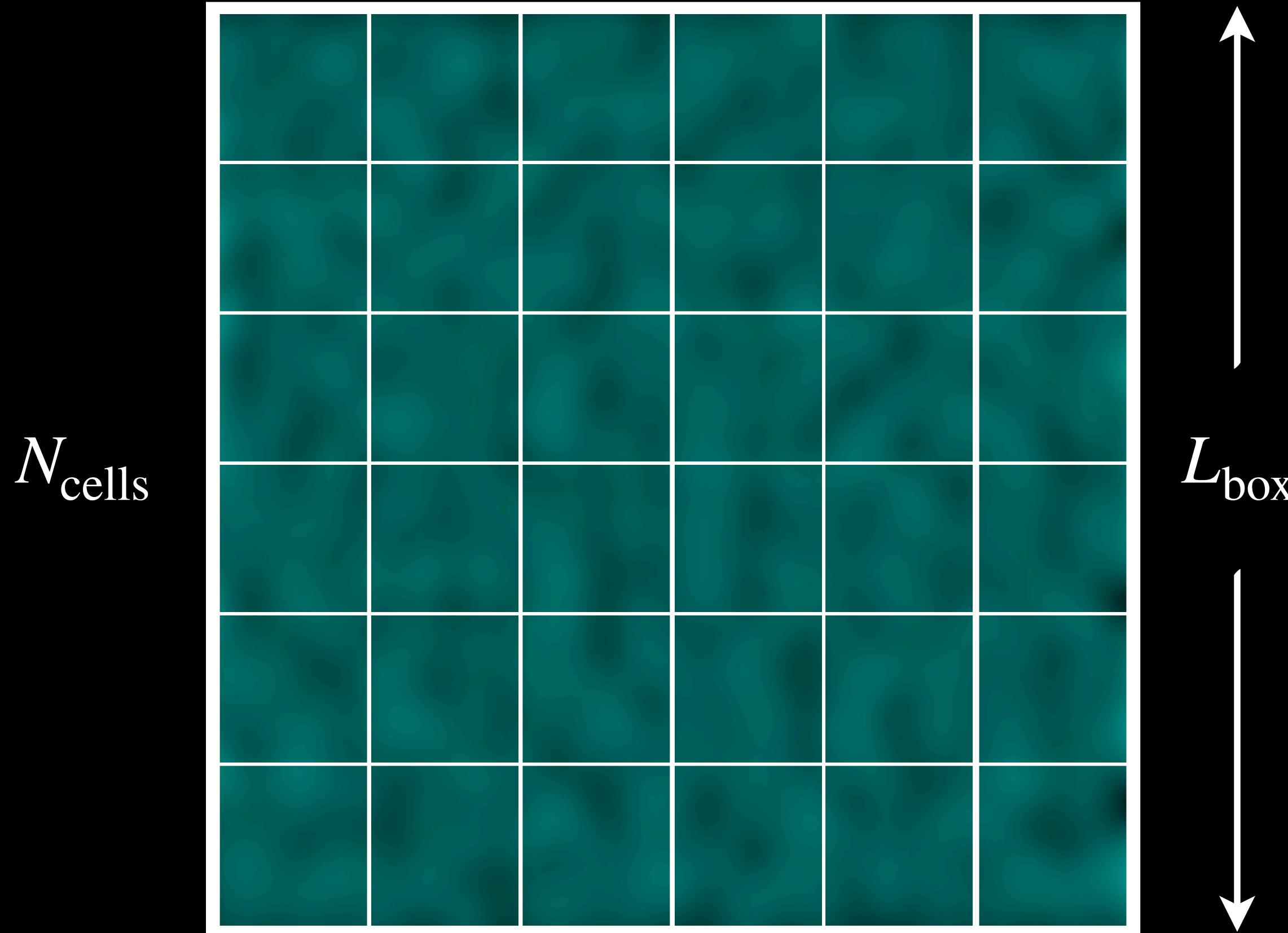


1. divide computational volume into a grid of size  $N_{\text{cells}} \times N_{\text{cells}} \times N_{\text{cells}}$
2. smooth the discrete particle distribution onto mesh using an interpolation scheme (CIC, NGP, TSC etc): density field  $\rho(\mathbf{x})$

use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# particle-mesh method

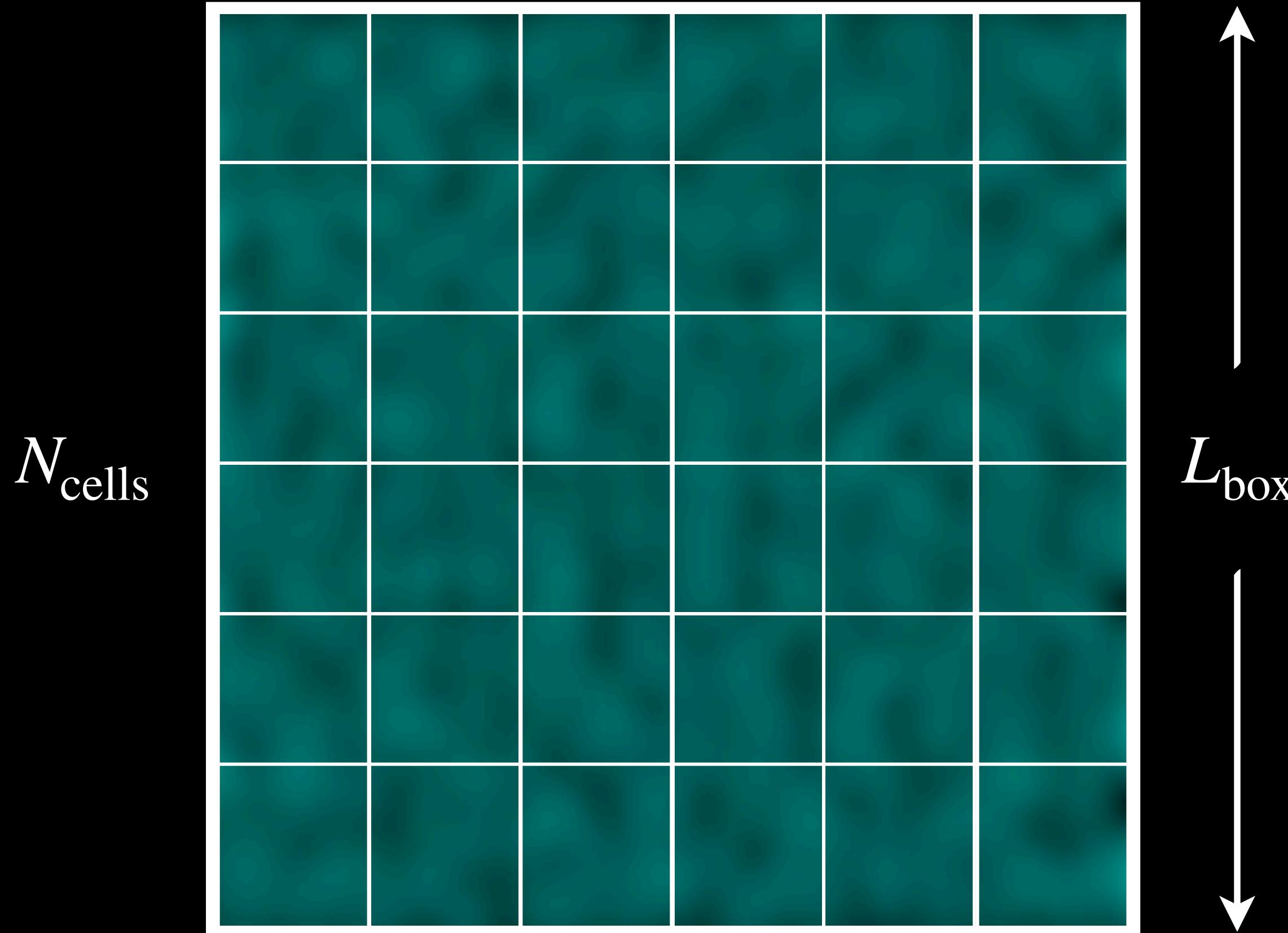


1. divide computational volume into a grid of size  $N_{\text{cells}} \times N_{\text{cells}} \times N_{\text{cells}}$
2. smooth the discrete particle distribution onto mesh using an interpolation scheme (CIC, NGP, TSC etc): density field  $\rho(\mathbf{x})$
3. Fourier transform density field:  $\rho(\mathbf{x}) \rightarrow \rho(\mathbf{k})$

use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# particle-mesh method

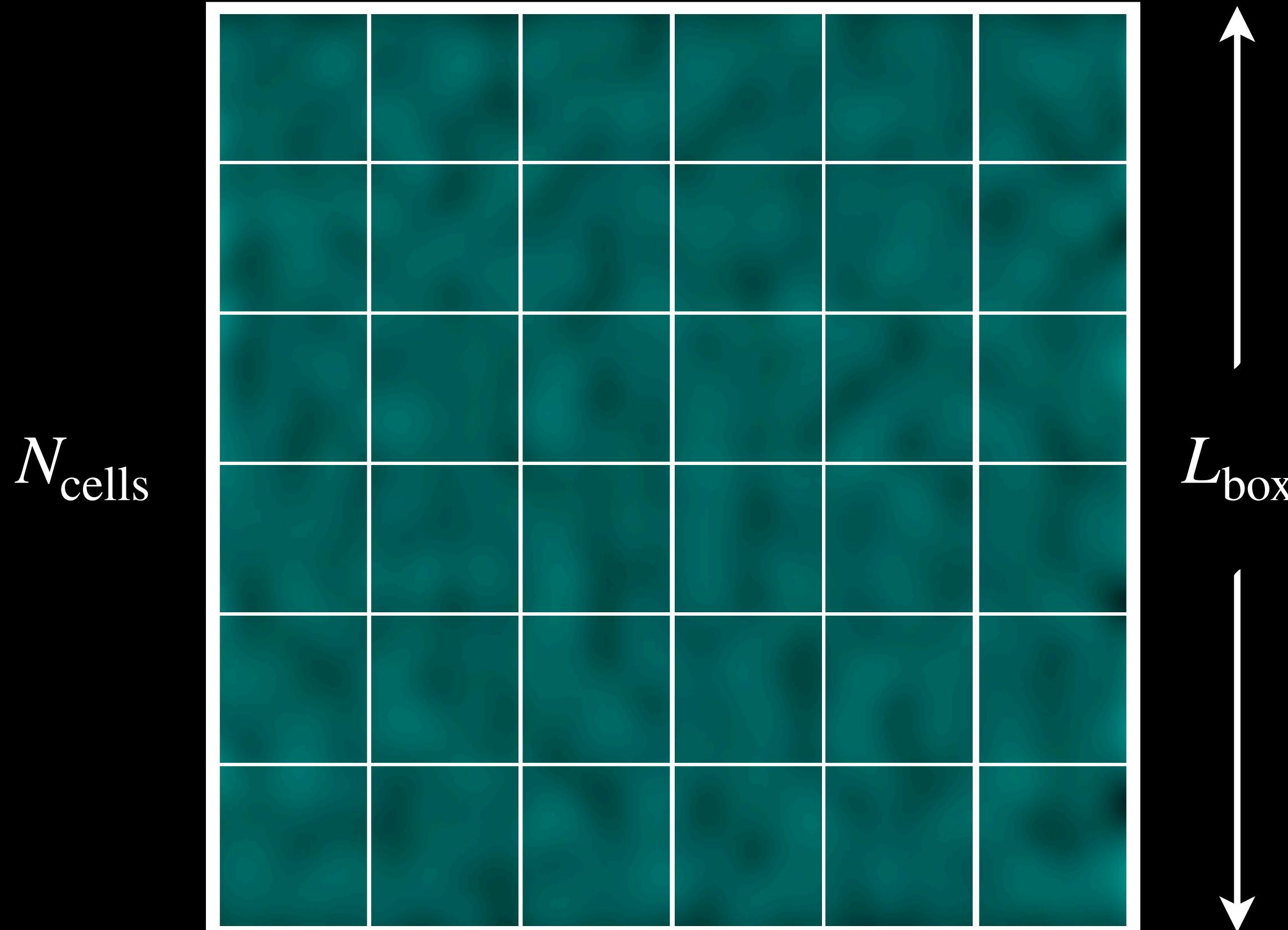


1. divide computational volume into a grid of size  $N_{\text{cells}} \times N_{\text{cells}} \times N_{\text{cells}}$
2. smooth the discrete particle distribution onto mesh using an interpolation scheme (CIC, NGP, TSC etc): density field  $\rho(\mathbf{x})$
3. Fourier transform density field:  $\rho(\mathbf{x}) \rightarrow \rho(\mathbf{k})$
4. solve Poisson's equation to get  $\Phi(\mathbf{k})$

use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# particle-mesh method



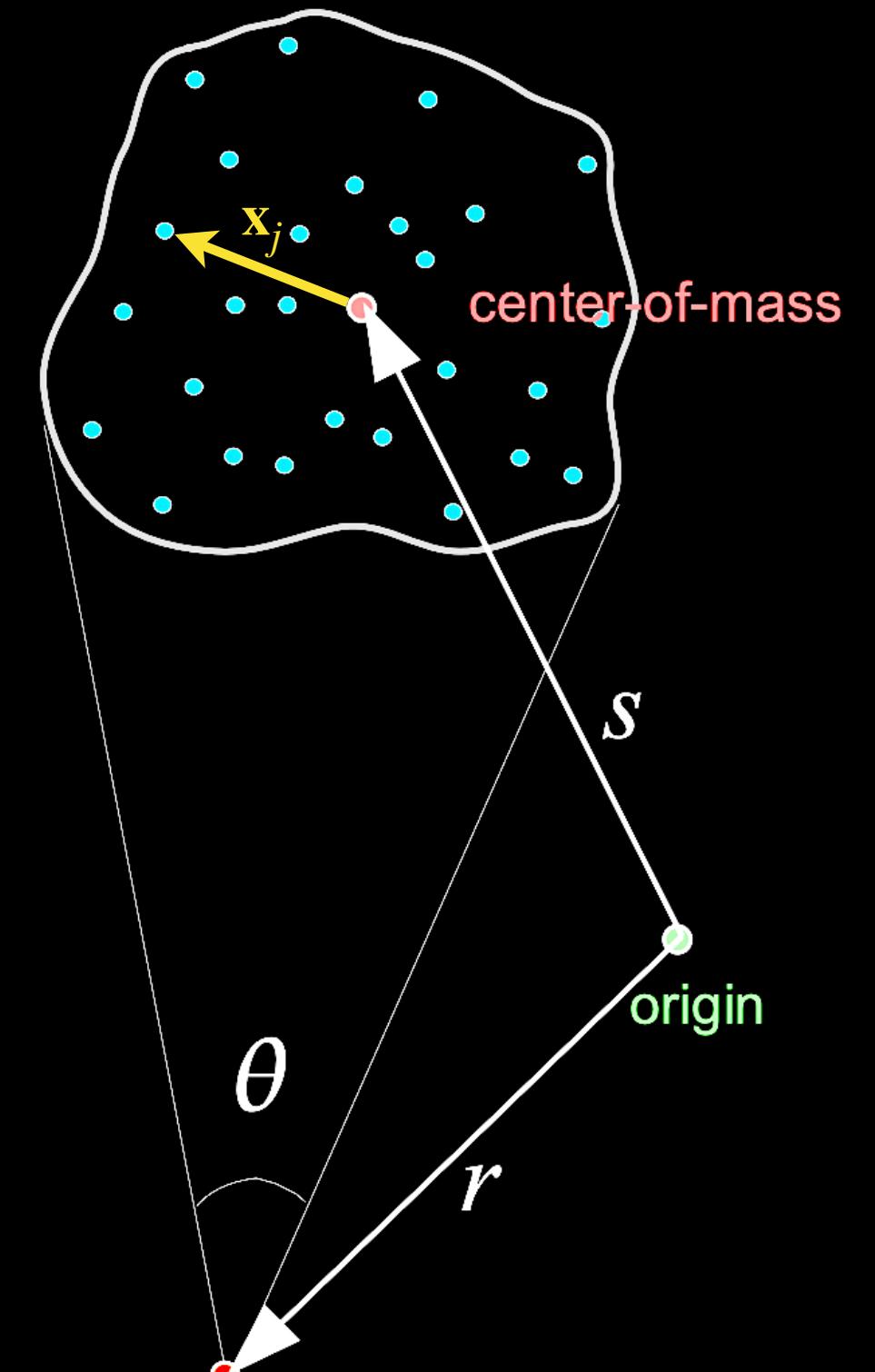
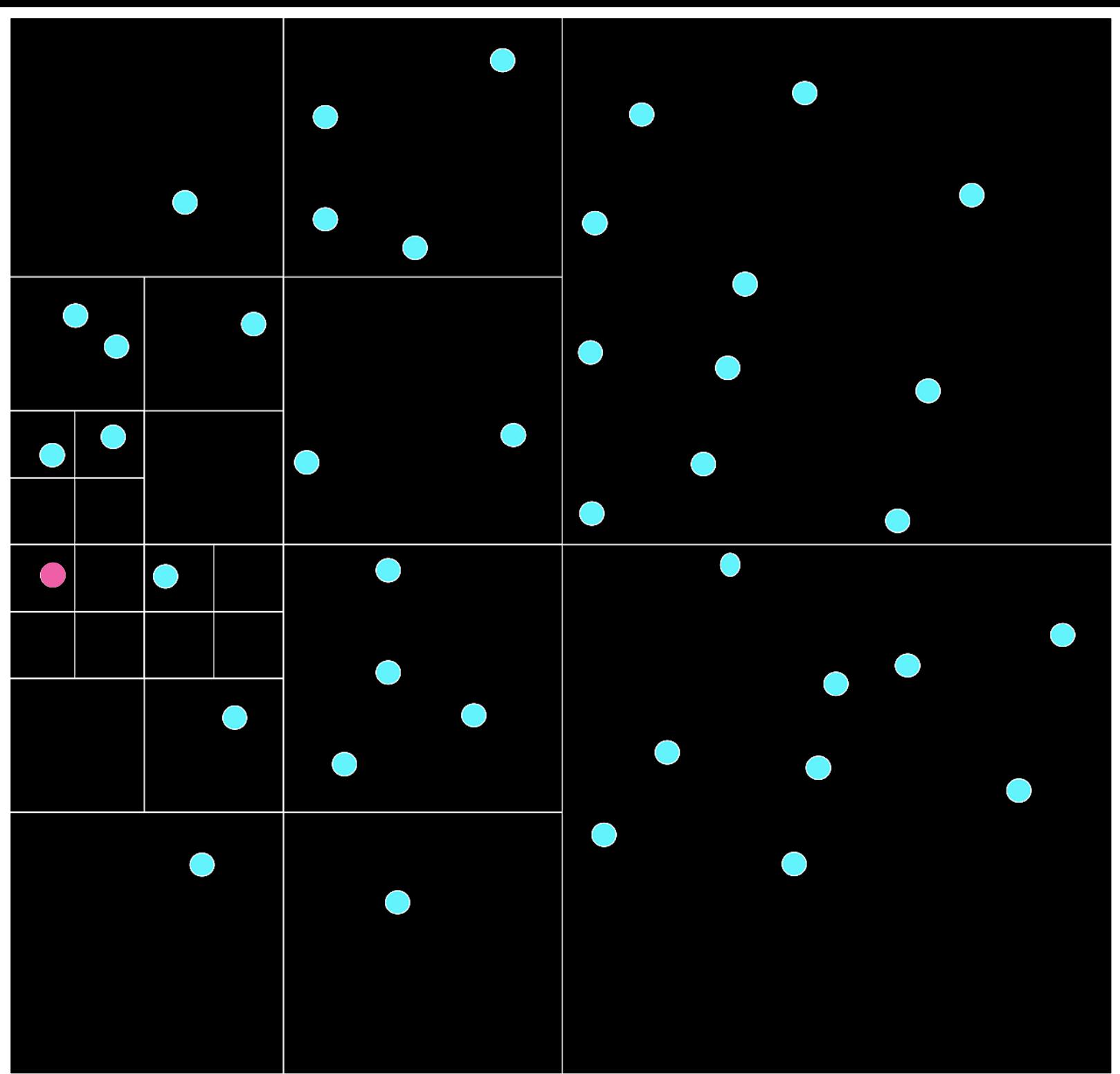
1. divide computational volume into a grid of size  $N_{\text{cells}} \times N_{\text{cells}} \times N_{\text{cells}}$
2. smooth the discrete particle distribution onto mesh using an interpolation scheme (CIC, NGP, TSC etc): density field  $\rho(\mathbf{x})$
3. Fourier transform density field:  $\rho(\mathbf{x}) \rightarrow \rho(\mathbf{k})$
4. solve Poisson's equation to get  $\Phi(\mathbf{k})$
5. inverse Fourier transform to get  $\Phi(\mathbf{x})$

use the properties of Fourier transforms + periodic boundary conditions to solve:

$$\Phi(\mathbf{k}) = G(\mathbf{k}) \cdot \rho(\mathbf{k})$$

# tree algorithms

based on the idea that you  
can group together particles  
that are a long way away  
from the particle of interest  
(pink)

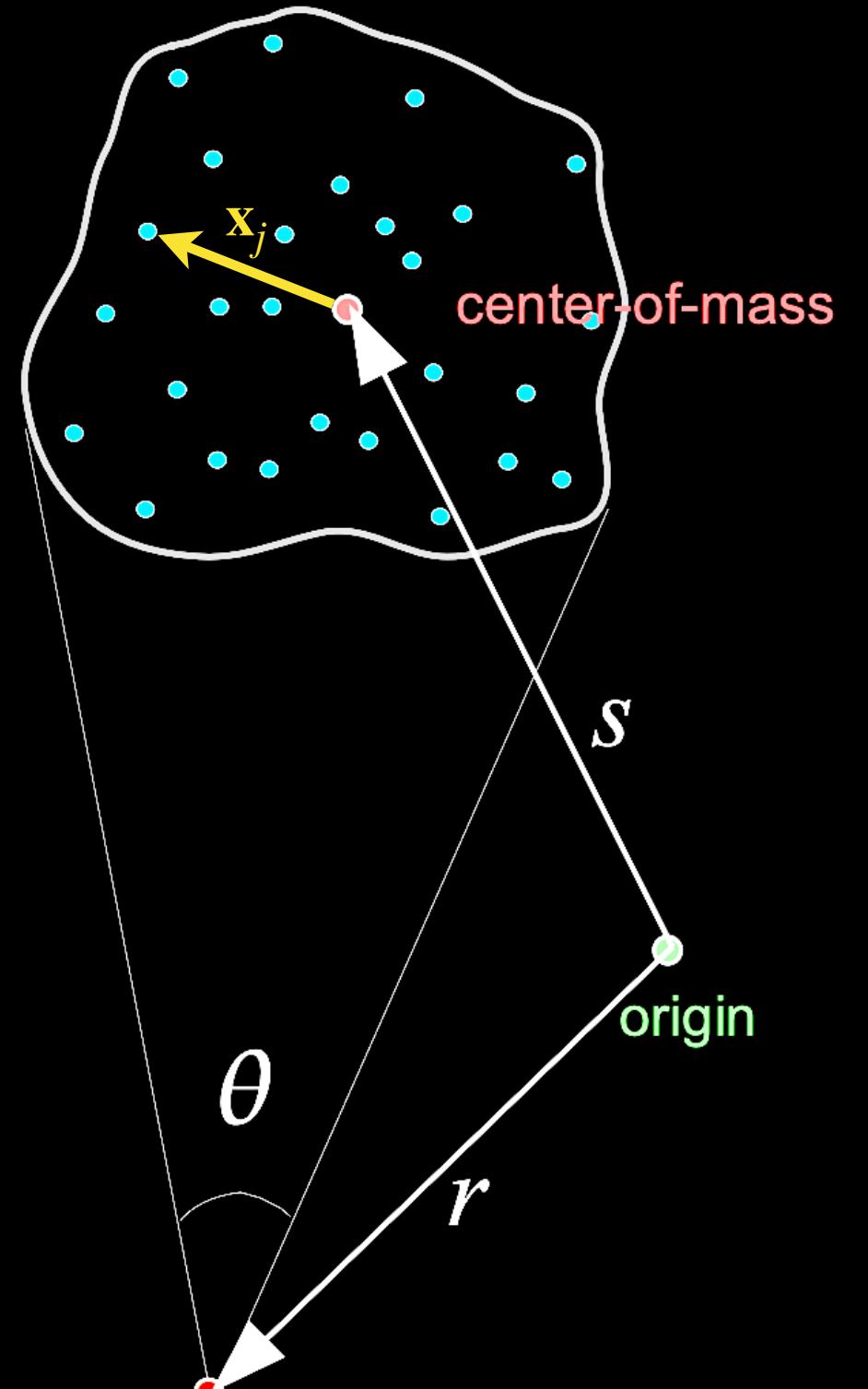
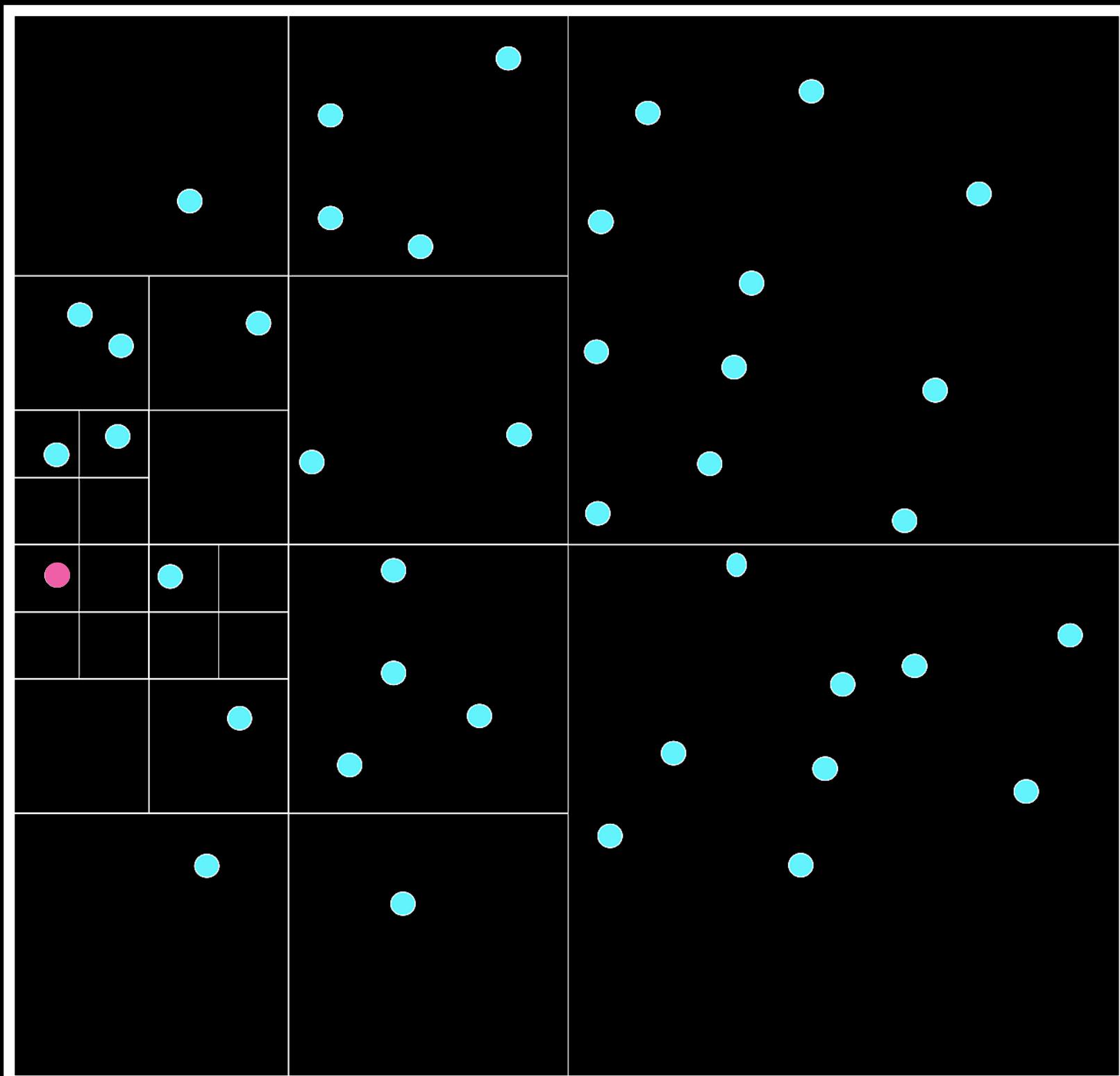


Springel (2005)

# tree algorithms

based on the idea that you can group together particles that are a long way away from the particle of interest (pink)

if  $\theta < \theta_{\text{crit}}$  : treat particles as a group contributing a monopole term (+ higher order terms if desired)



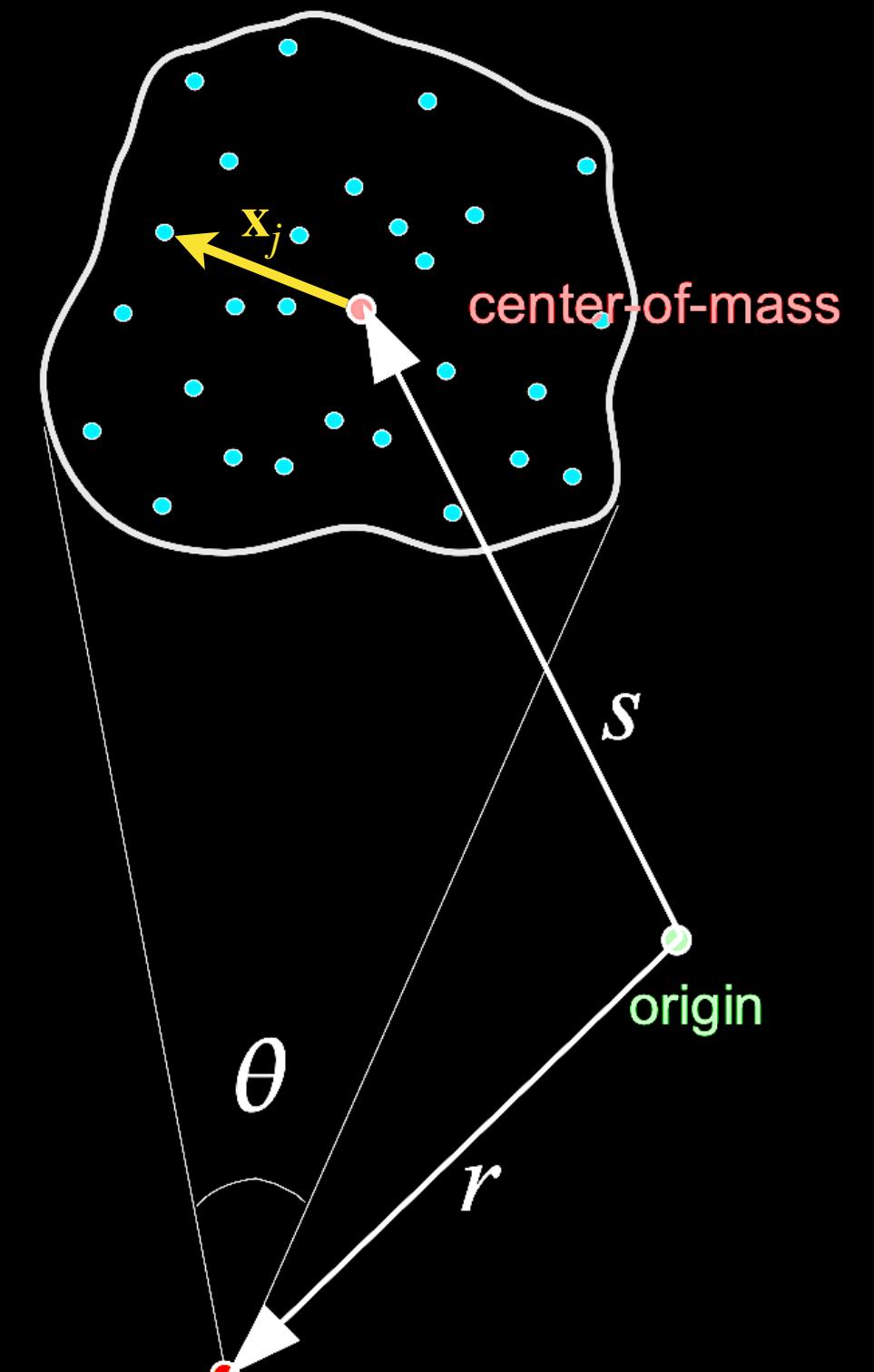
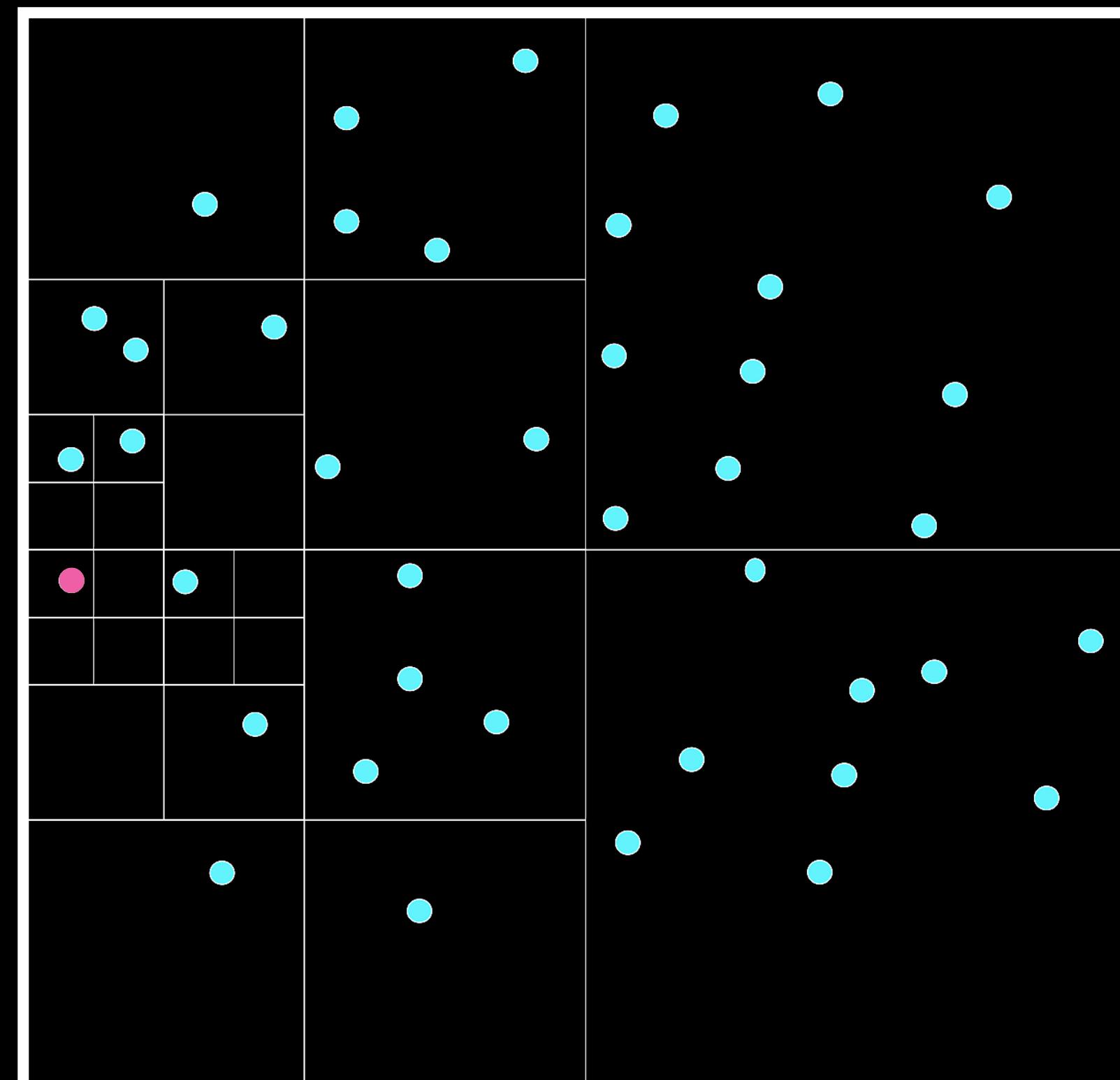
Springel (2005)

# tree algorithms

based on the idea that you can group together particles that are a long way away from the particle of interest (pink)

if  $\theta < \theta_{\text{crit}}$  : treat particles as a group contributing a monopole term (+ higher order terms if desired)

if  $\theta > \theta_{\text{crit}}$  : open up parent cell and process sub-cells individually



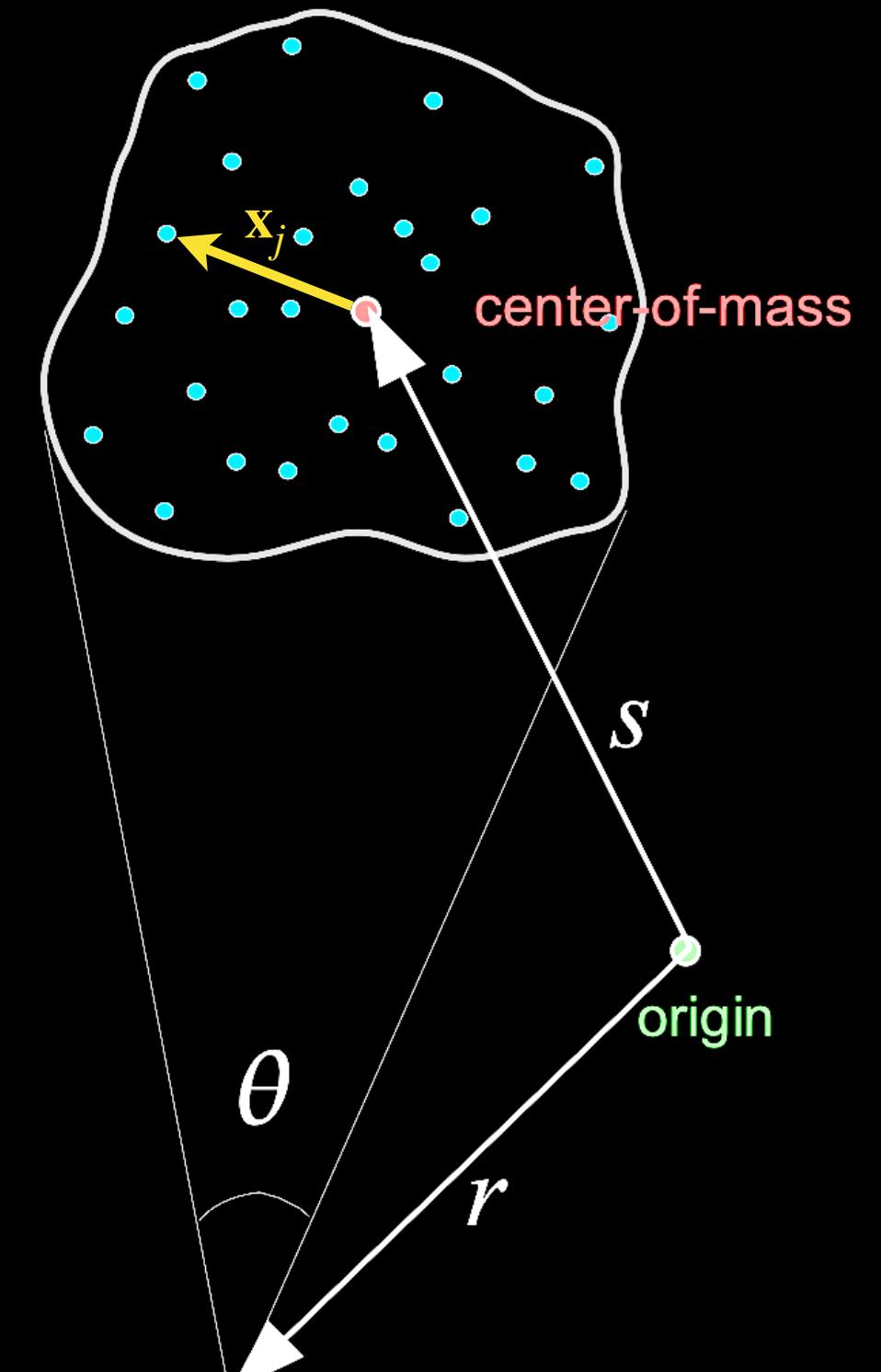
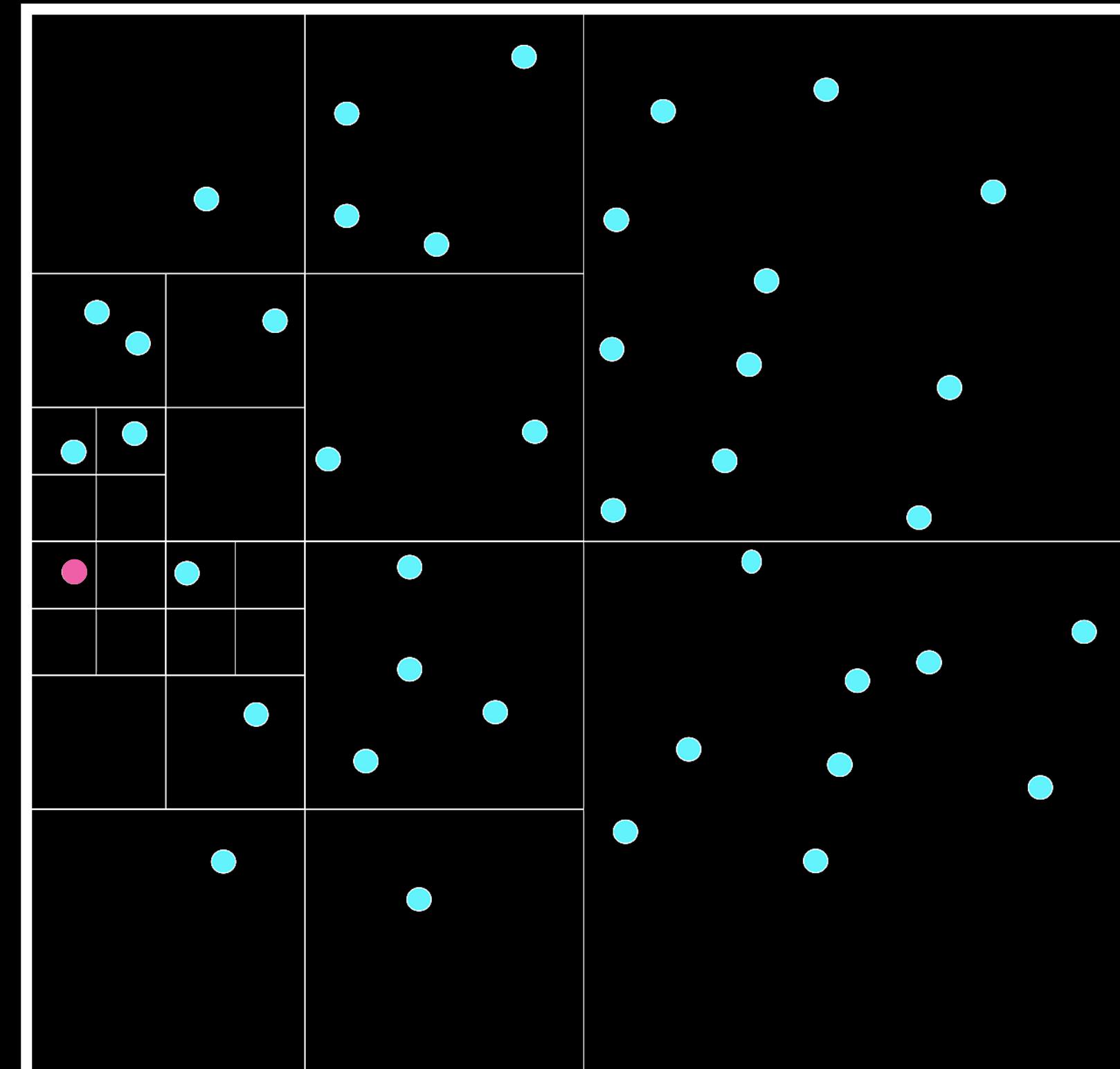
Springel (2005)

# tree algorithms

based on the idea that you can group together particles that are a long way away from the particle of interest (pink)

if  $\theta < \theta_{\text{crit}}$  : treat particles as a group contributing a monopole term (+ higher order terms if desired)

if  $\theta > \theta_{\text{crit}}$  : open up parent cell and process sub-cells individually

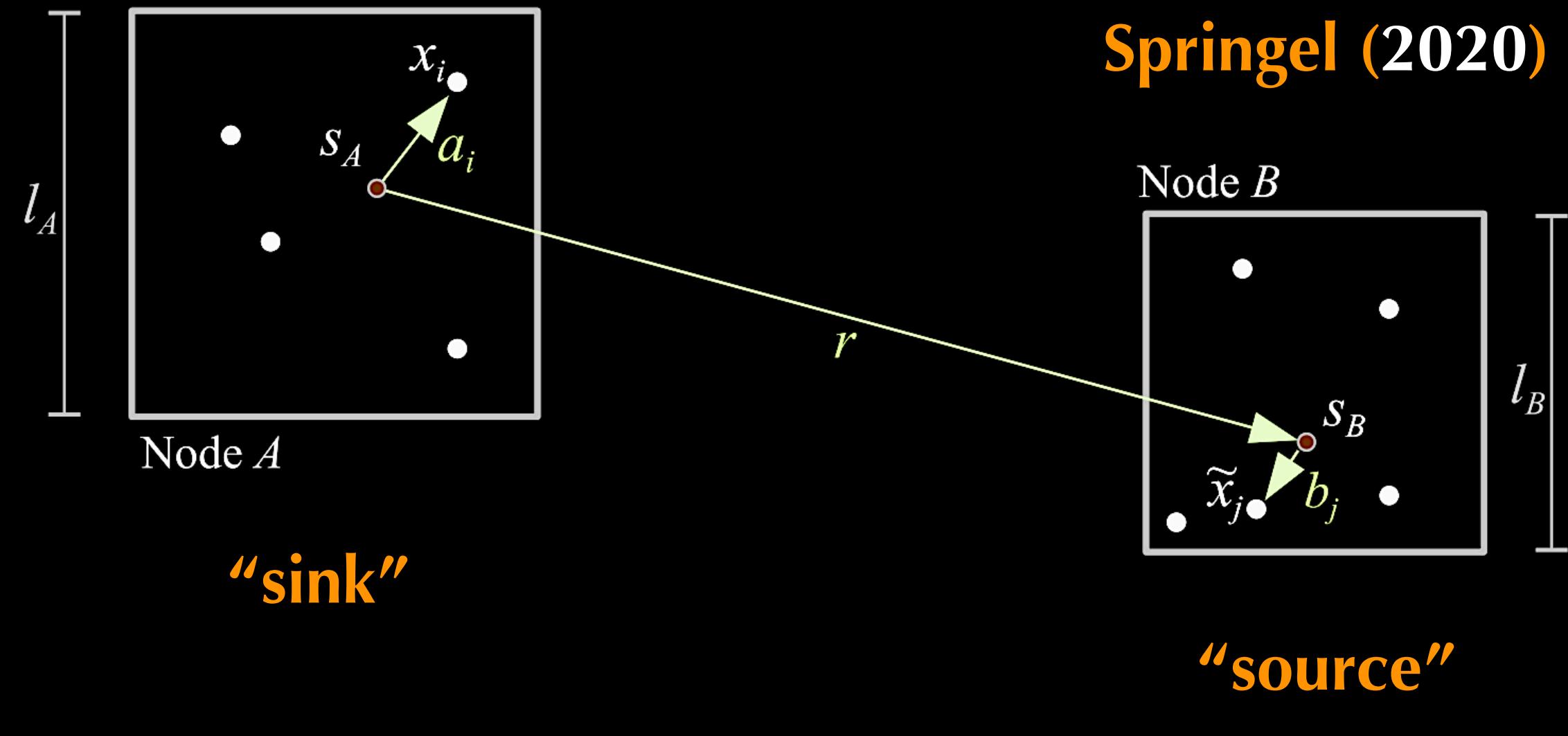


Springel (2005)

gravitational potential generated by points inside a node:

$$\Phi(\mathbf{x}) = -G \sum_{j \in \text{node}} m_j g \left( \tilde{\mathbf{x}}_j - \mathbf{x} \right)$$

# fast multipole method



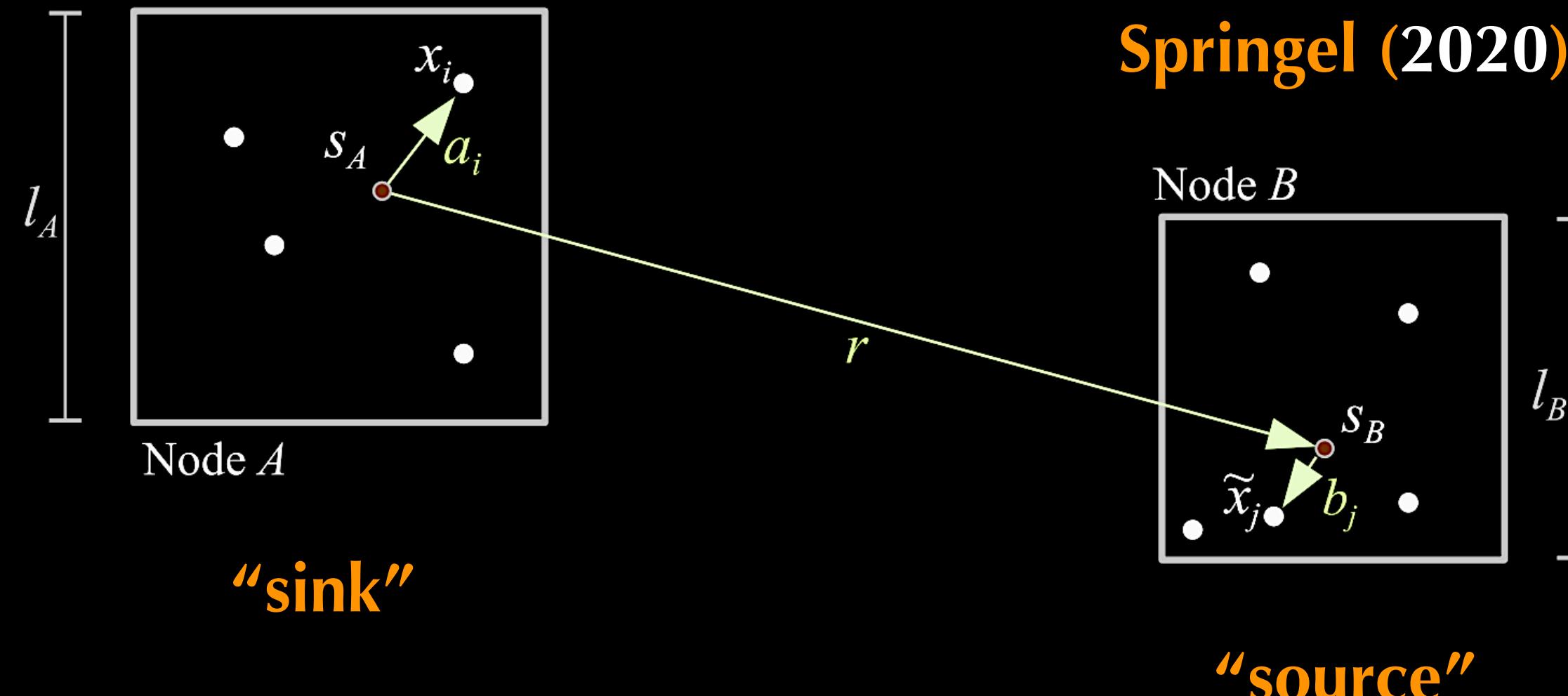
Springel (2020)

the fast multipole method (FMM) is arguably  
the most efficient way to compute  
hierarchical multipole expansions of the  
gravitational potential with high accuracy.

⇒ perform multipole expansion of  
potential on both the “source” and “sink”  
sides just once — this can then be reused  
for all particles in each node

$$\Phi(\mathbf{x}_i) = -G \sum_j m_j g(\tilde{\mathbf{x}}_j - \mathbf{x}_i)$$

# fast multipole method



Springel (2020)

the fast multipole method (FMM) is arguably  
the most efficient way to compute  
hierarchical multipole expansions of the  
gravitational potential with high accuracy.

⇒ perform multipole expansion of  
potential on both the “source” and “sink”  
sides just once — this can then be reused  
for all particles in each node

$$\Phi(\mathbf{x}_i) = -G \sum_j m_j g(\tilde{\mathbf{x}}_j - \mathbf{x}_i)$$

$$g(\tilde{\mathbf{x}}_j - \mathbf{x}_i) \simeq \sum_{n=0}^p \frac{1}{n!} \nabla_{\mathbf{r}}^{(n)} g(\mathbf{r}) \cdot (\mathbf{b}_j - \mathbf{a}_i)^{(n)}$$

higher-order multipole  
expansions ⇒ more  
accurate forces

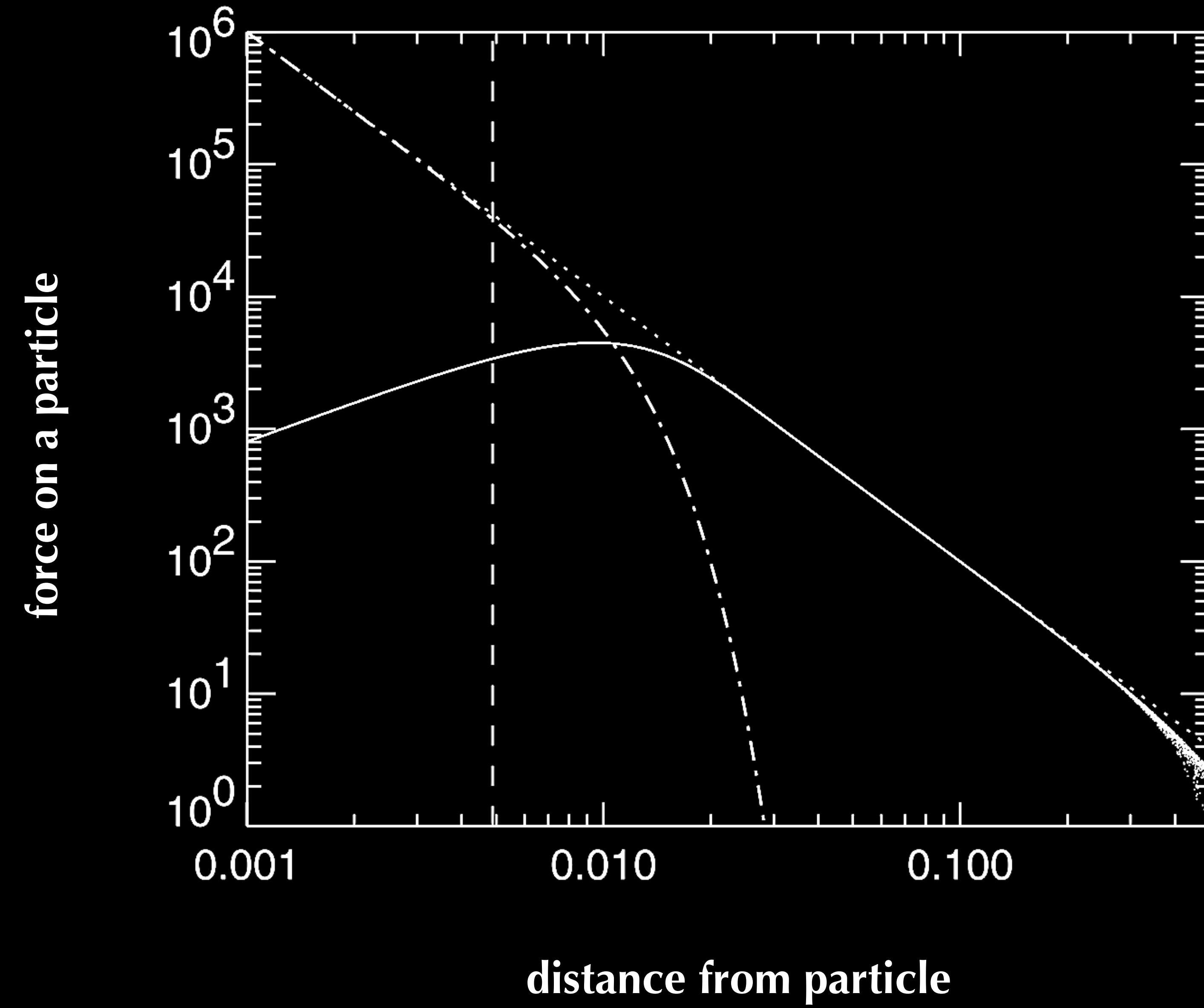
$p = 1$  (dipole)

$p = 2$  (quadrupole)

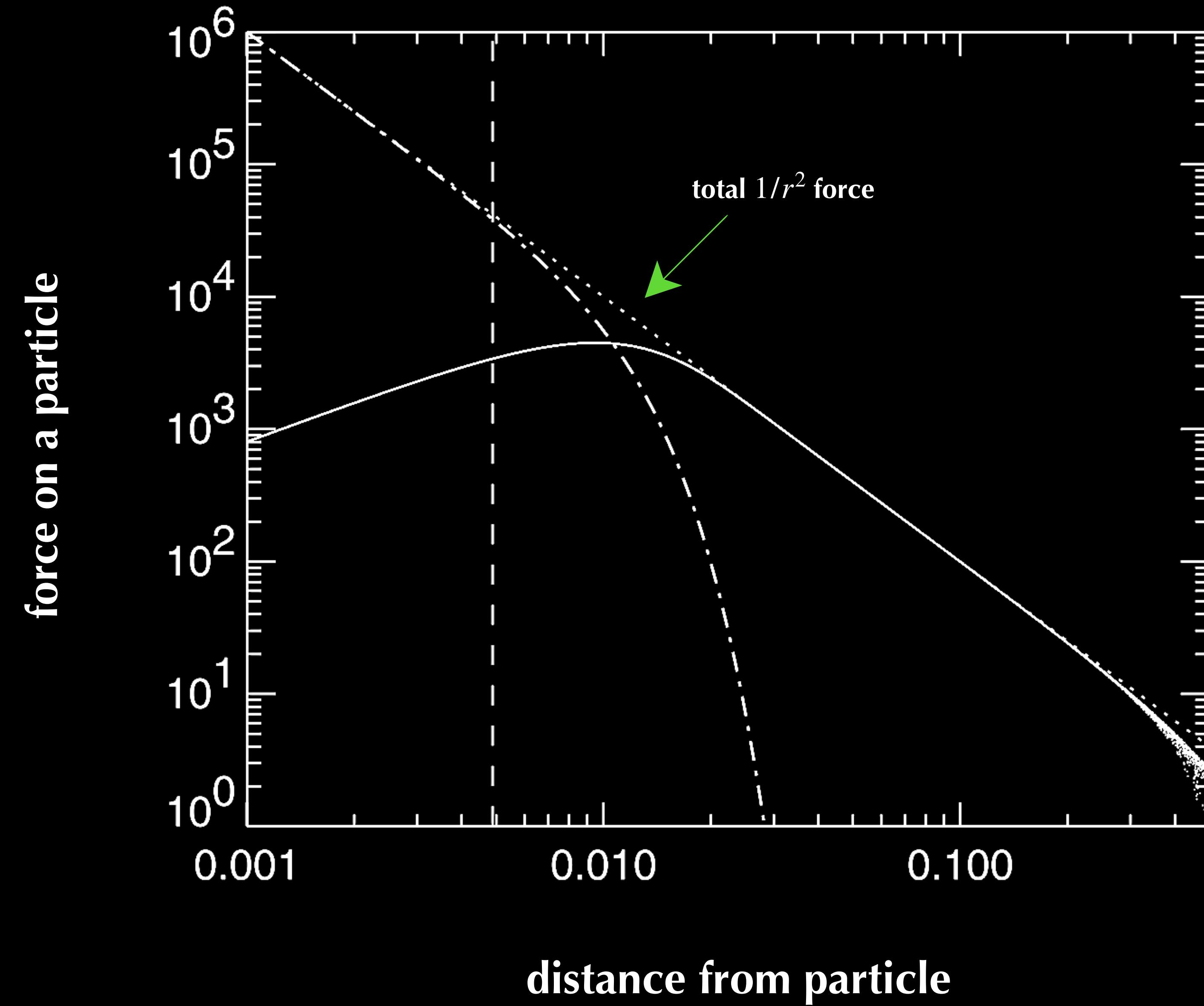
$p = 3$  (octupole)

$p = 4$  (hexadecapole)

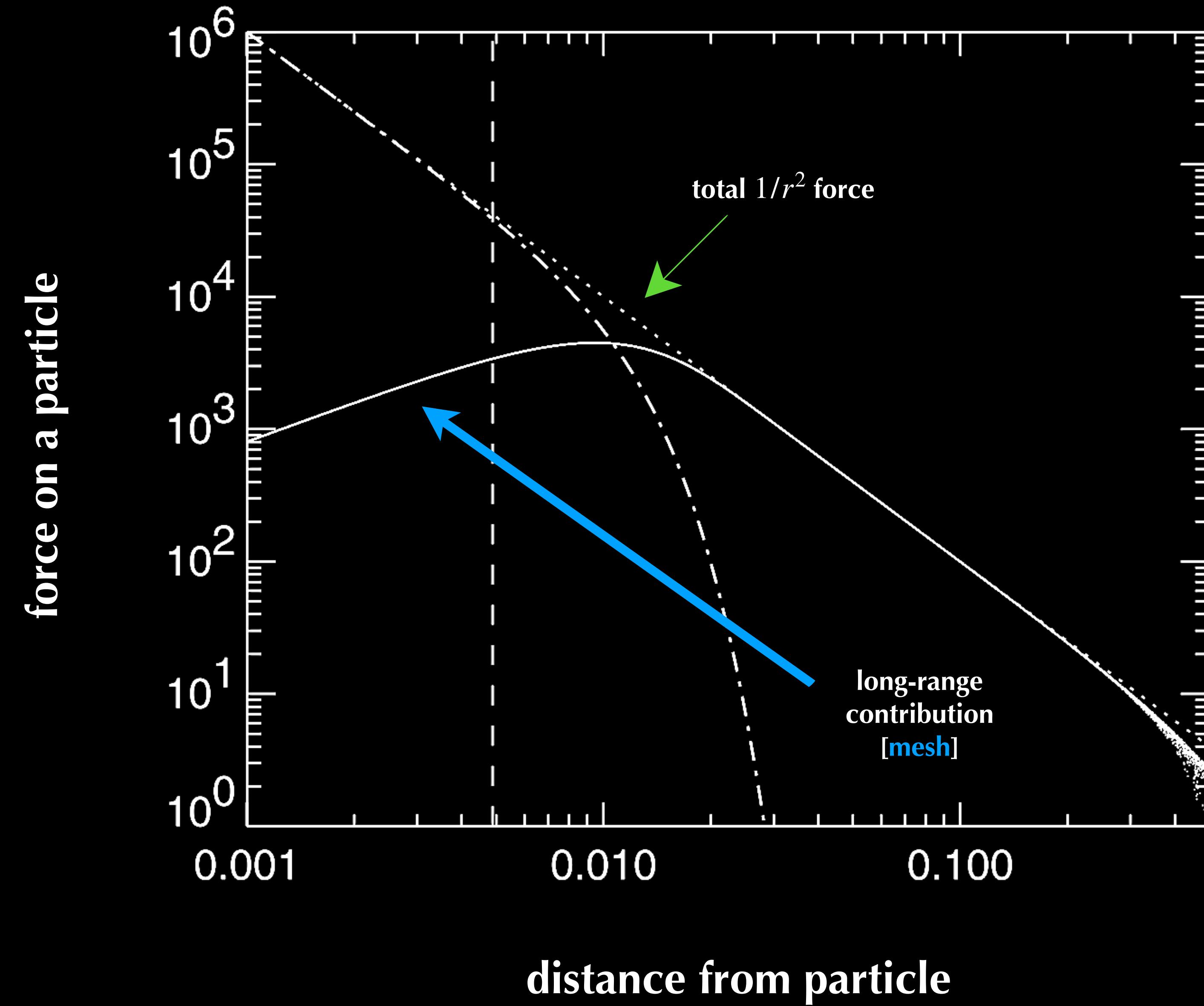
$p = 5$  (triakontadipole)



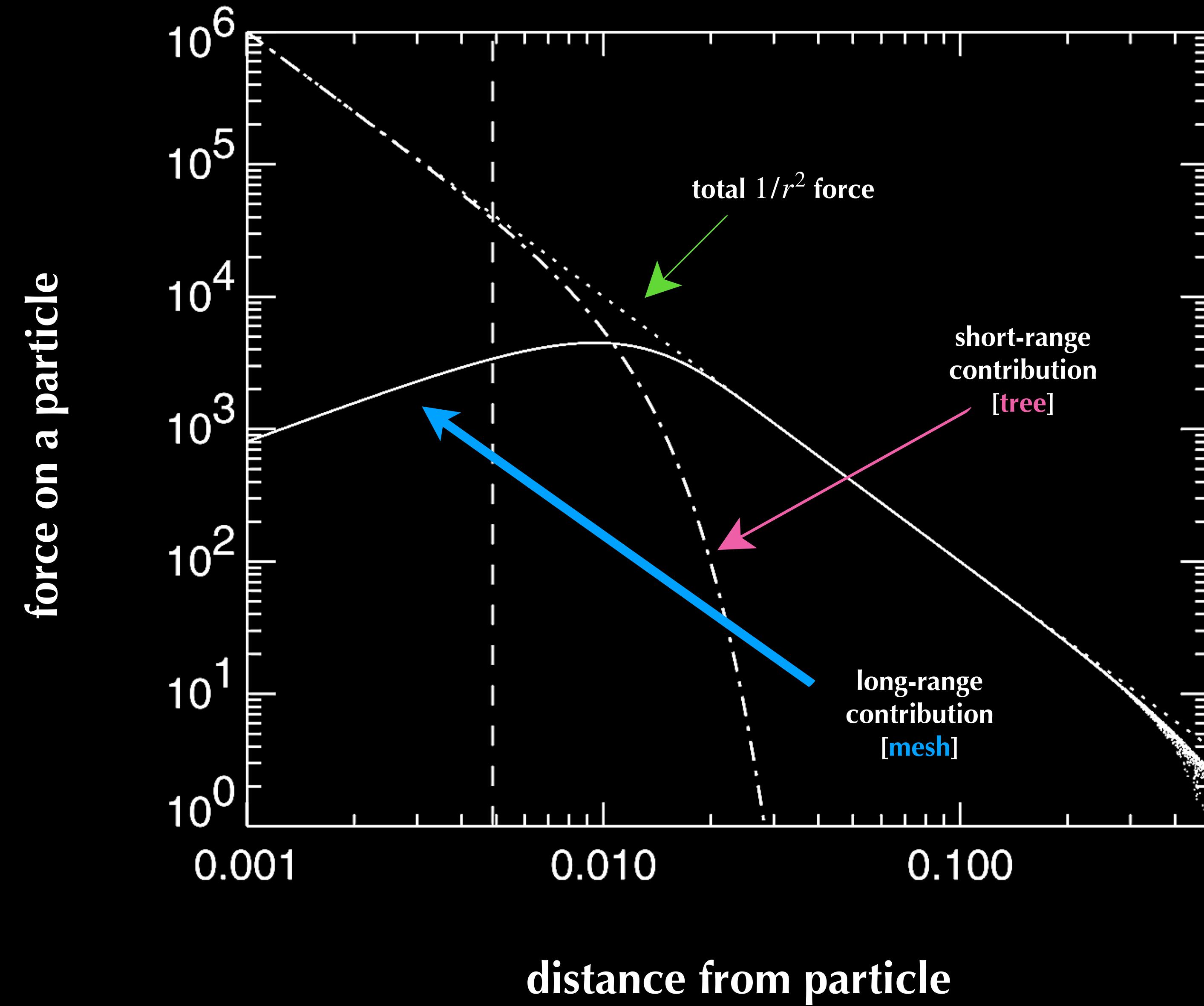
Springel (2005)



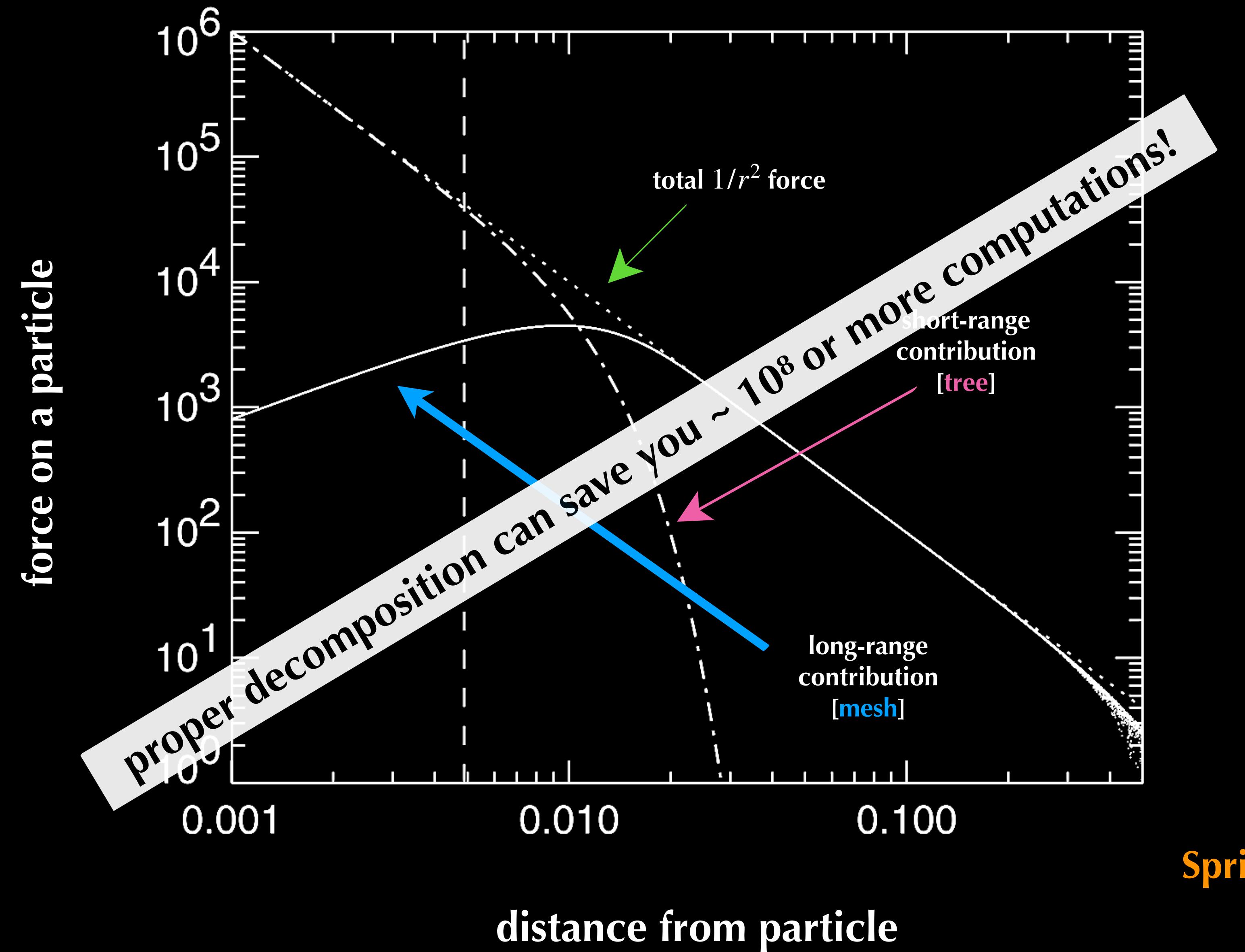
Springel (2005)



Springel (2005)



Springel (2005)



Springel (2005)

# basic time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

# basic time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^n \cdot \Delta t$$

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^n + \mathbf{a}_j^n \cdot \Delta t$$

simplest case: the Euler method

simple

only first-order accurate

# basic time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

acceleration  
[computed from before]

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^n \cdot \Delta t$$

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^n + \underline{\mathbf{a}_j^n} \cdot \Delta t$$

simplest case: the Euler method

simple  
only first-order accurate

# basic time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

$$\mathbf{X}_j^{n+1} = \mathbf{X}_j^n + \mathbf{V}_j^n \cdot \Delta t$$
$$\mathbf{V}_j^{n+1} = \mathbf{V}_j^n + \mathbf{a}_j^n \cdot \Delta t$$

acceleration  
[computed from before]

timestep

simplest case: the Euler method

simple  
only first-order accurate

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

$$\begin{aligned}\mathbf{v}_j^{n+1/2} &= \mathbf{v}_j^n + \frac{1}{2} \mathbf{a}_j^n \cdot \Delta t \\ \mathbf{x}_j^{n+1} &= \mathbf{x}_j^n + \mathbf{v}_j^{n+1/2} \cdot \Delta t \\ \mathbf{v}_j^{n+1} &= \mathbf{v}_j^{n+1/2} + \frac{1}{2} \mathbf{a}_j^{n+1} \cdot \Delta t\end{aligned}$$

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

kick

$$\mathbf{v}_j^{n+1/2} = \mathbf{v}_j^n + \frac{1}{2} \mathbf{a}_j^n \cdot \Delta t$$

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^{n+1/2} \cdot \Delta t$$

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^{n+1/2} + \frac{1}{2} \mathbf{a}_j^{n+1} \cdot \Delta t$$

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

kick

$$\mathbf{v}_j^{n+1/2} = \mathbf{v}_j^n + \frac{1}{2} \mathbf{a}_j^n \cdot \Delta t$$

drift

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^{n+1/2} \cdot \Delta t$$

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^{n+1/2} + \frac{1}{2} \mathbf{a}_j^{n+1} \cdot \Delta t$$

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

kick

$$\mathbf{v}_j^{n+1/2} = \mathbf{v}_j^n + \frac{1}{2} \mathbf{a}_j^n \cdot \Delta t$$

drift

$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^{n+1/2} \cdot \Delta t$$

kick

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^{n+1/2} + \frac{1}{2} \mathbf{a}_j^{n+1} \cdot \Delta t$$

# higher-order time integration

once forces on particles have been calculated, we need to propagate these forward in time and update their (1) **positions** and (2) **velocities**

kick

$$\mathbf{v}_j^{n+1/2} = \mathbf{v}_j^n + \frac{1}{2} \mathbf{a}_j^n \cdot \Delta t$$

drift

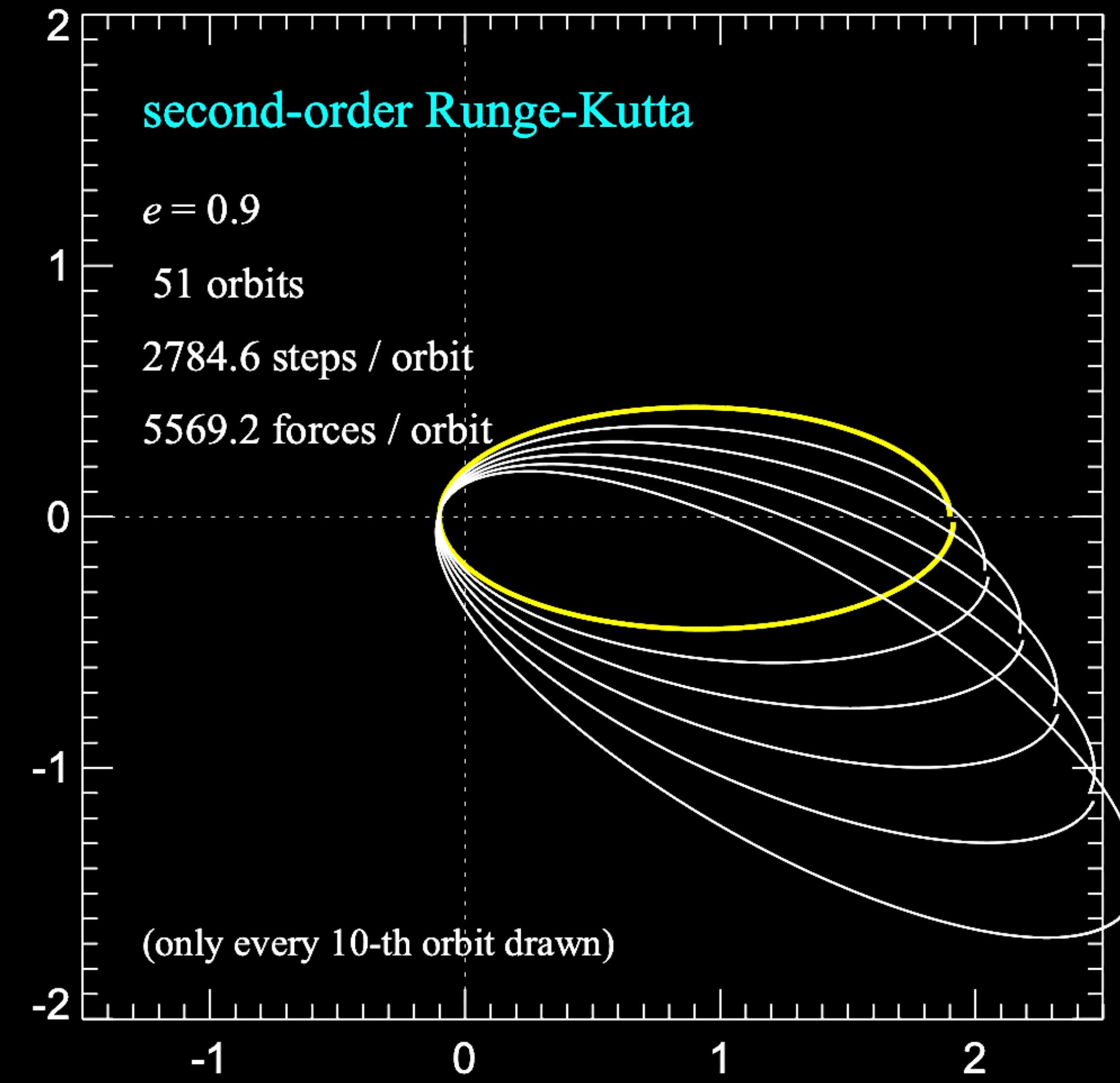
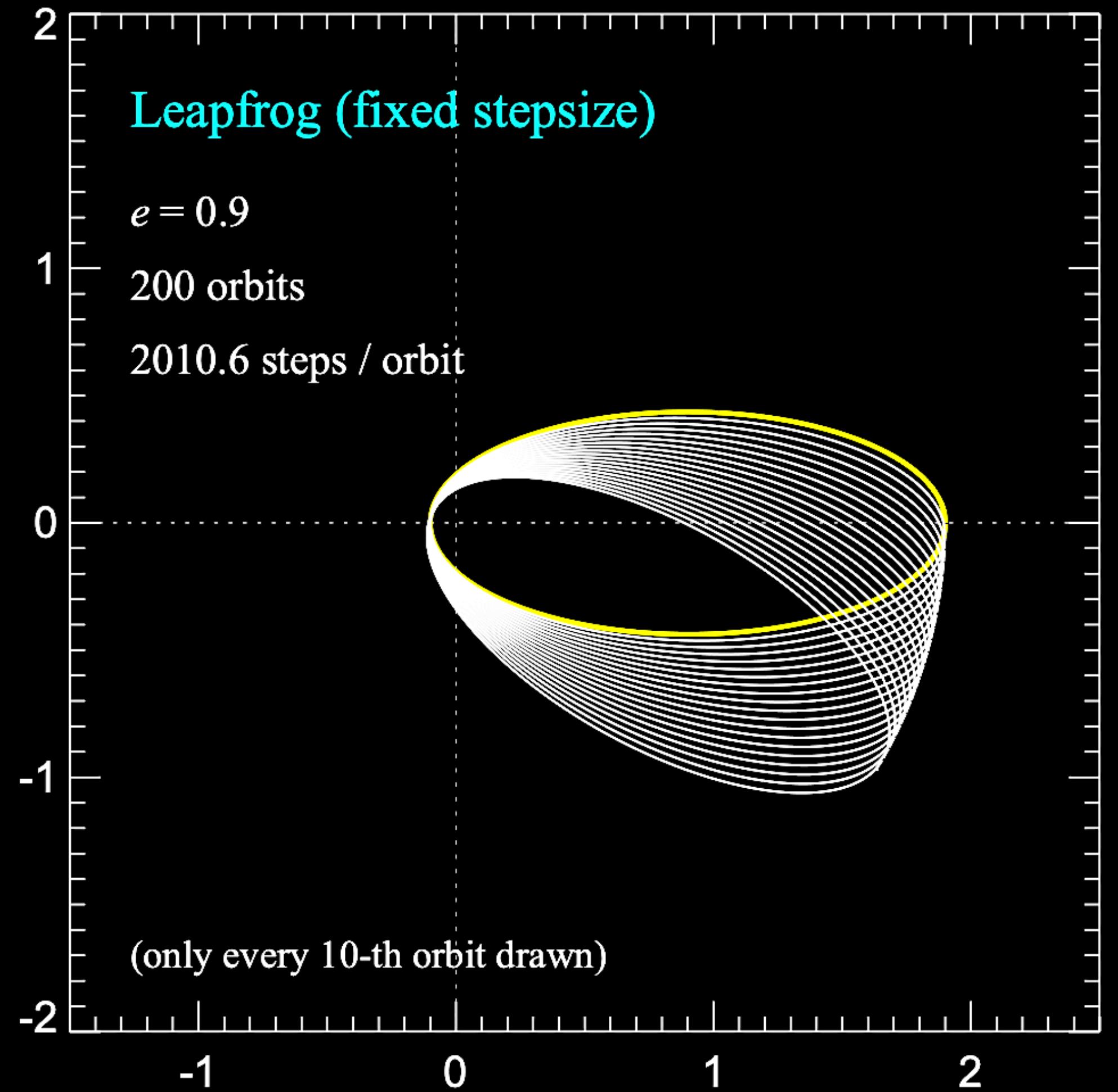
$$\mathbf{x}_j^{n+1} = \mathbf{x}_j^n + \mathbf{v}_j^{n+1/2} \cdot \Delta t$$

kick

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^{n+1/2} + \frac{1}{2} \mathbf{a}_j^{n+1} \cdot \Delta t$$

2nd order  
“leapfrog” method

much more stable  
“symplectic”  
[~energy is conserved]



**Springel (2005)**

# choice of timestep

ideally, one would want to choose the size of the timestep,  $\Delta t$ , to be as small as possible

this is impractical from a computational standpoint, but also unnecessary: close-range particle pairs need to be integrated on much shorter timesteps than distant pairs. solution: adopt a hierarchy of timesteps determined by the force acting on the particle

$$\Delta t_j = \eta \sqrt{\frac{\epsilon}{|\mathbf{a}_j|}}$$

# choice of timestep

ideally, one would want to choose the size of the timestep,  $\Delta t$ , to be as small as possible

this is impractical from a computational standpoint, but also unnecessary: close-range particle pairs need to be integrated on much shorter timesteps than distant pairs. solution: adopt a hierarchy of timesteps determined by the force acting on the particle

$$\Delta t_j = \eta \sqrt{\frac{\epsilon}{|\mathbf{a}_j|}}$$

integration “tolerance” parameter

**lecture notes:**

<https://github.com/sownakbose/AstroPGCourse-Nbody>

**jupyter notebooks:**

[https://github.com/Shaun-T-Brown/Summer\\_school](https://github.com/Shaun-T-Brown/Summer_school)  
[link also on lecture notes Github page]

