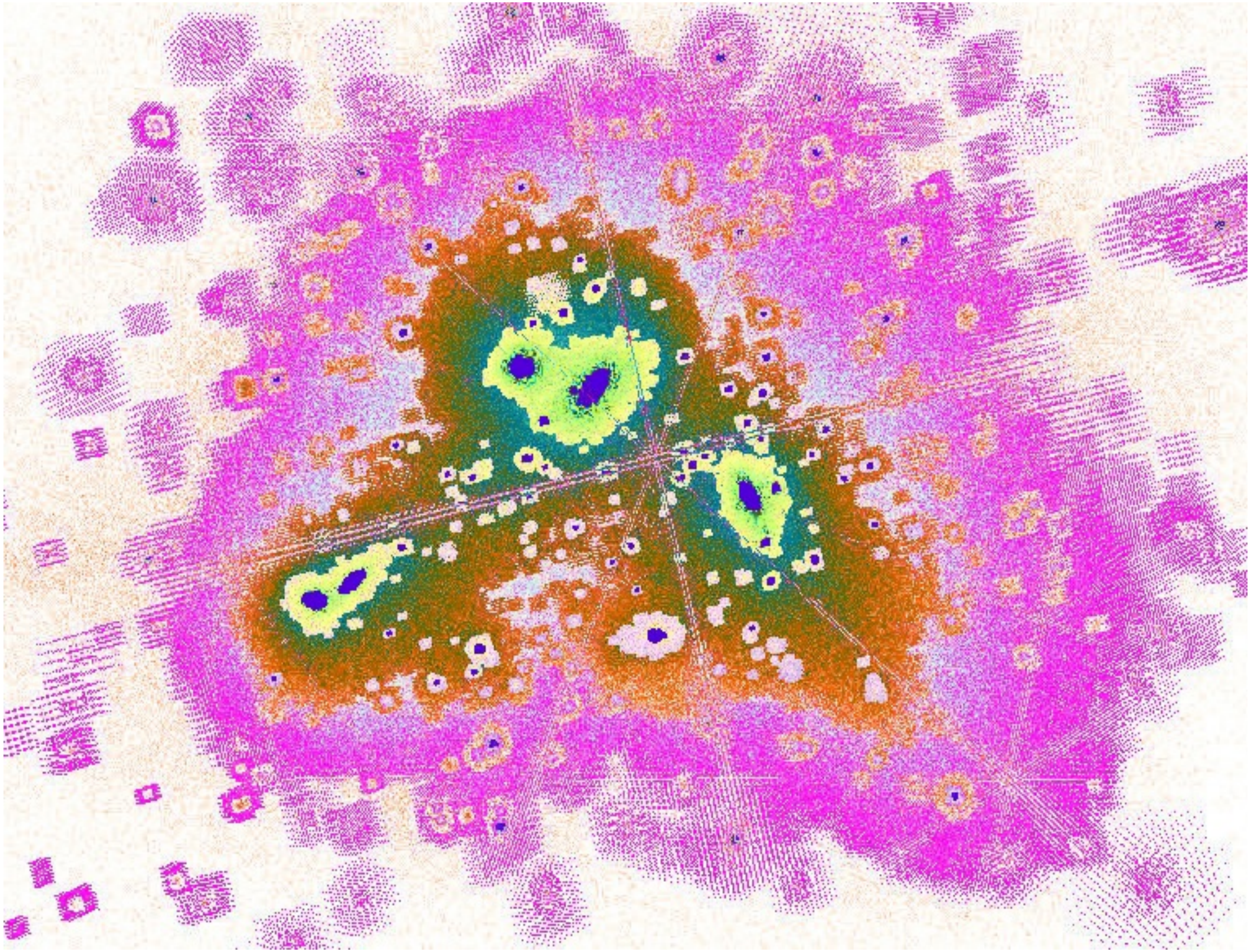
AHF - Amiga's Halo Finder

User's Guide

built 299



Alexander Knebe & Steffen Knollmann, December 2009

DISCLAIMER

All Software described in this User's Guide and provided for download comes with no warranty or guarantee to function!

Further, this Guide does not claim to be complete; it has been compiled to the best knowledge and primarily lists those options and features that are considered "useful" for the general black-box user...

The proper references for all things ***AHF*** are the code papers

Gill S.P.D., Knebe A., Gibson B.K., 2004, MNRAS, 351, 399
Knollmann S.R., Knebe A., 2009, ApJS, 182,608

Please refer to these publications for more information and the relevant tests and please
cite them both when publishing results based upon ***AHF***.

USER'S GUIDE

- INTRODUCTION
- CONCEPT
- HOW TO COMPILE? (**DEFINEFLAGS**)
- HOW TO RUN?
- SUPPORTED INPUT FILE FORMATS
- FORMAT OF THE OUTPUT FILES
- **TOOLBOX:**
  - **MERGERTREE**
  - **HALOTRACKER**

Introduction

- **MLAPM** (**M**ulti-**L**evel-**A**daptive-**P**article-**M**esh)

when	what	who
1997	grid structure	Andrew Green
2000	complete revision	Alexander Knebe
2001	public release	Knebe, Green & Binney (2001)
2002	software package for lightcones	Enn Saar
2004	**MHF**: on-the-fly halo identification	Stuart Gill (Gill, Knebe & Gibson 2004)
2005	name change **MLAPM** -> ***AMIGA***	Alexander Knebe

- **MLAPM** (**M**ulti-**L**evel-**A**daptive-**P**article-**M**esh)

when	what	who
1997	grid structure	Andrew Green
2000	complete revision	Alexander Knebe
2001	public release	Knebe, Green & Binney (2001)
2002	software package for lightcones	Enn Saar
2004	**MHF**: on-the-fly halo identification	Stuart Gill (Gill, Knebe & Gibson 2004)
2005	name change **MLAPM** -> ***AMIGA***	Alexander Knebe

MLAPM's users-guide.pdf **already contains information about MHF!**

- ***AMIGA (Adaptive Mesh Investigations of Galaxy Assembly)***

when	what	who
2005	name change **MLAPM** -> ***AMIGA***	Alexander Knebe
	name change **MHF** -> ***AHF***	Alexander Knebe
2007	release of MPI enabled ***AHF***	Steffen Knollmann
2007+	revisions over revisions...	Alexander Knebe, Steffen Knollmann, Kristin Warnick, Claudio Llinares, ...

- ***AMIGA (Adaptive Mesh Investigations of Galaxy Assembly)***

when	what	who
2005	name change **MLAPM** -> ***AMIGA***	Alexander Knebe
	name change **MHF** -> ***AHF***	Alexander Knebe
2007	release of MPI enabled ***AHF***	Steffen Knollmann
2007+	revisions over revisions...	Alexander Knebe, Steffen Knollmann, Kristin Warnick, Claudio Llinares, ...

- **this document (hopefully) provides...**

✓ **explanation of all things *AHF***

✓ **introduction of bundled software packages:**

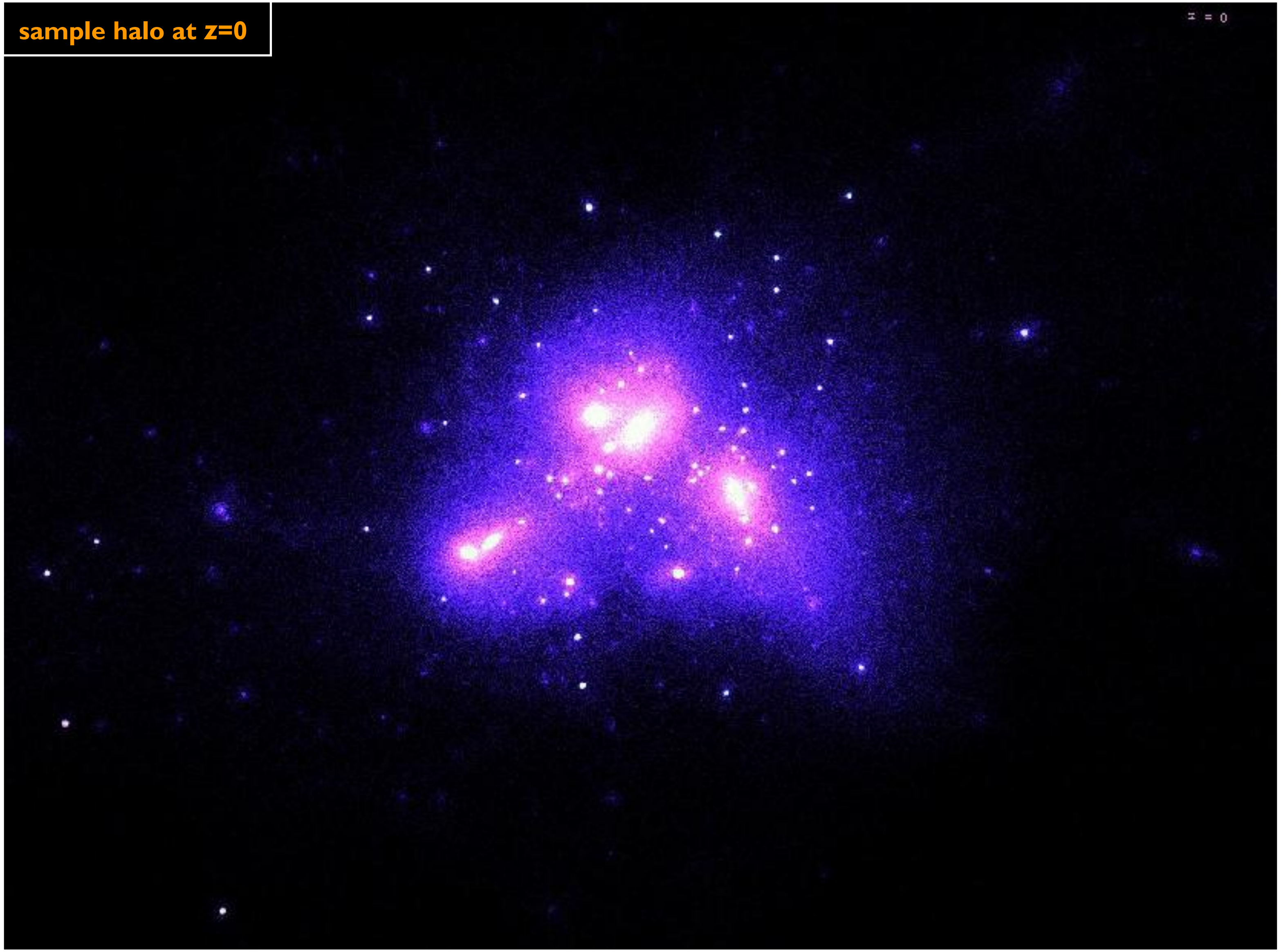
- **MergerTree.c**
- **HaloTracker.c**

Concept

CONCEPT

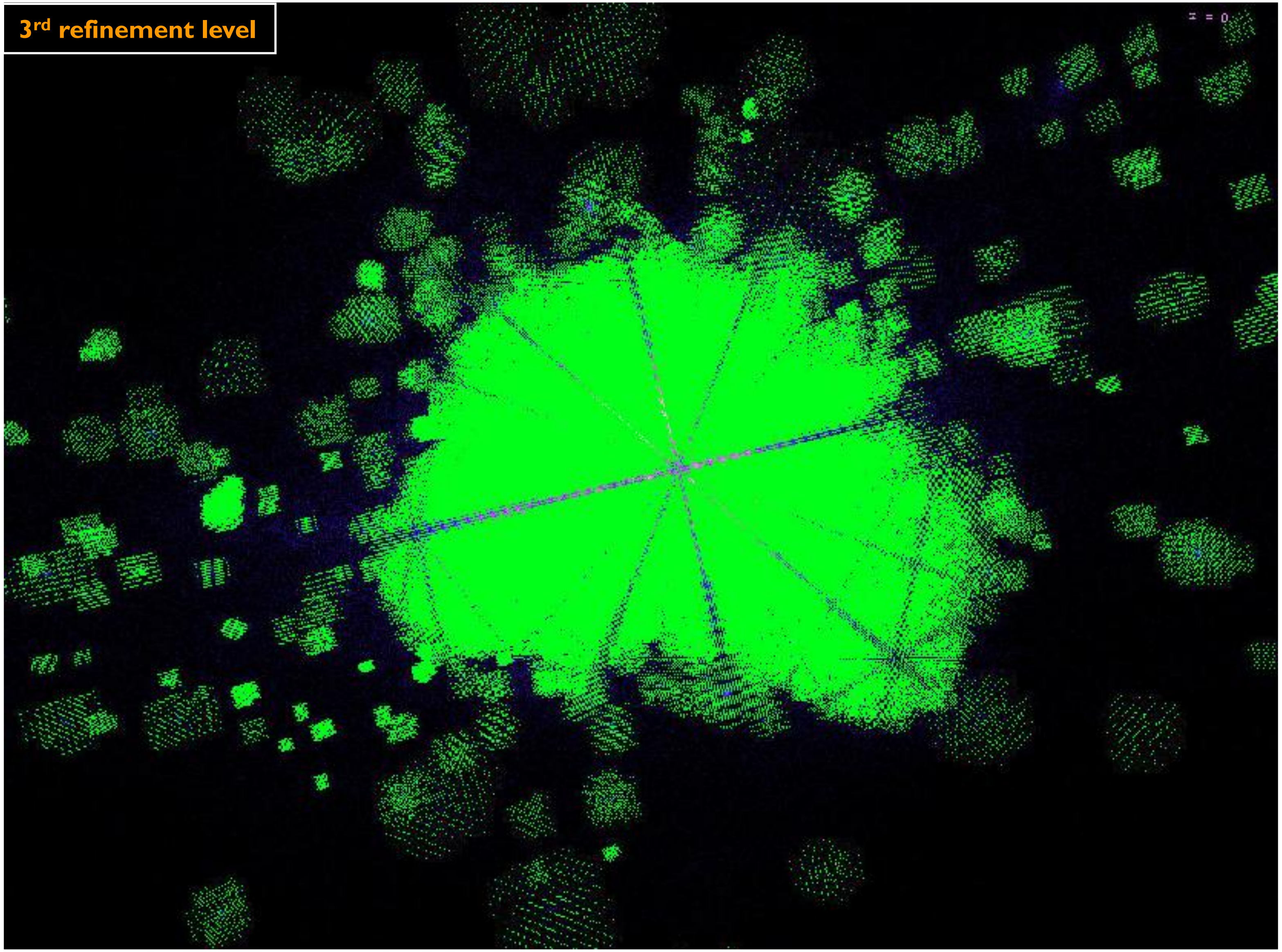
- finding prospective halo centres
- collecting particles possibly bound to centre
- removing unbound particles
- calculating halo properties

CONCEPT

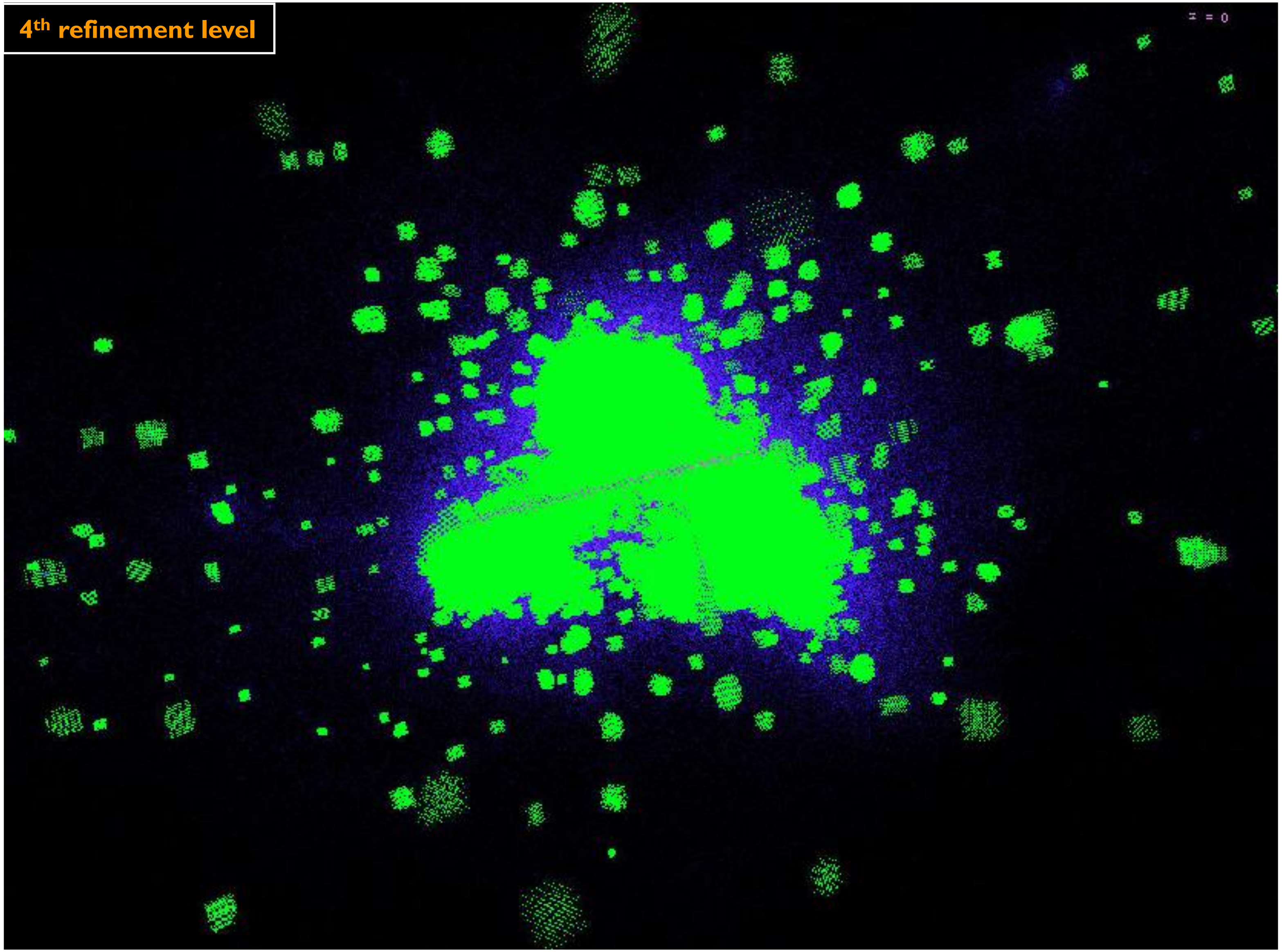


CONCEPT

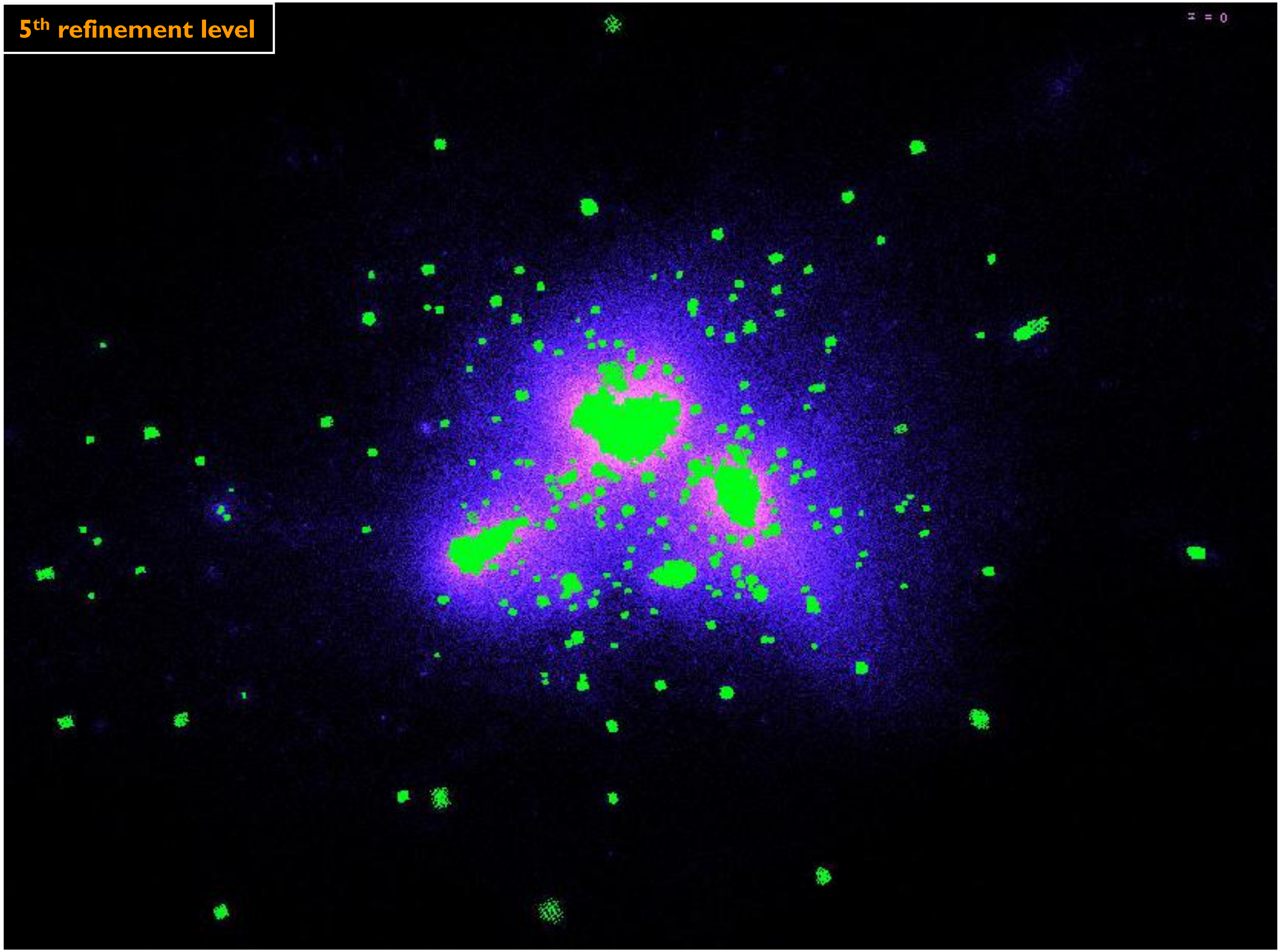
CONCEPT



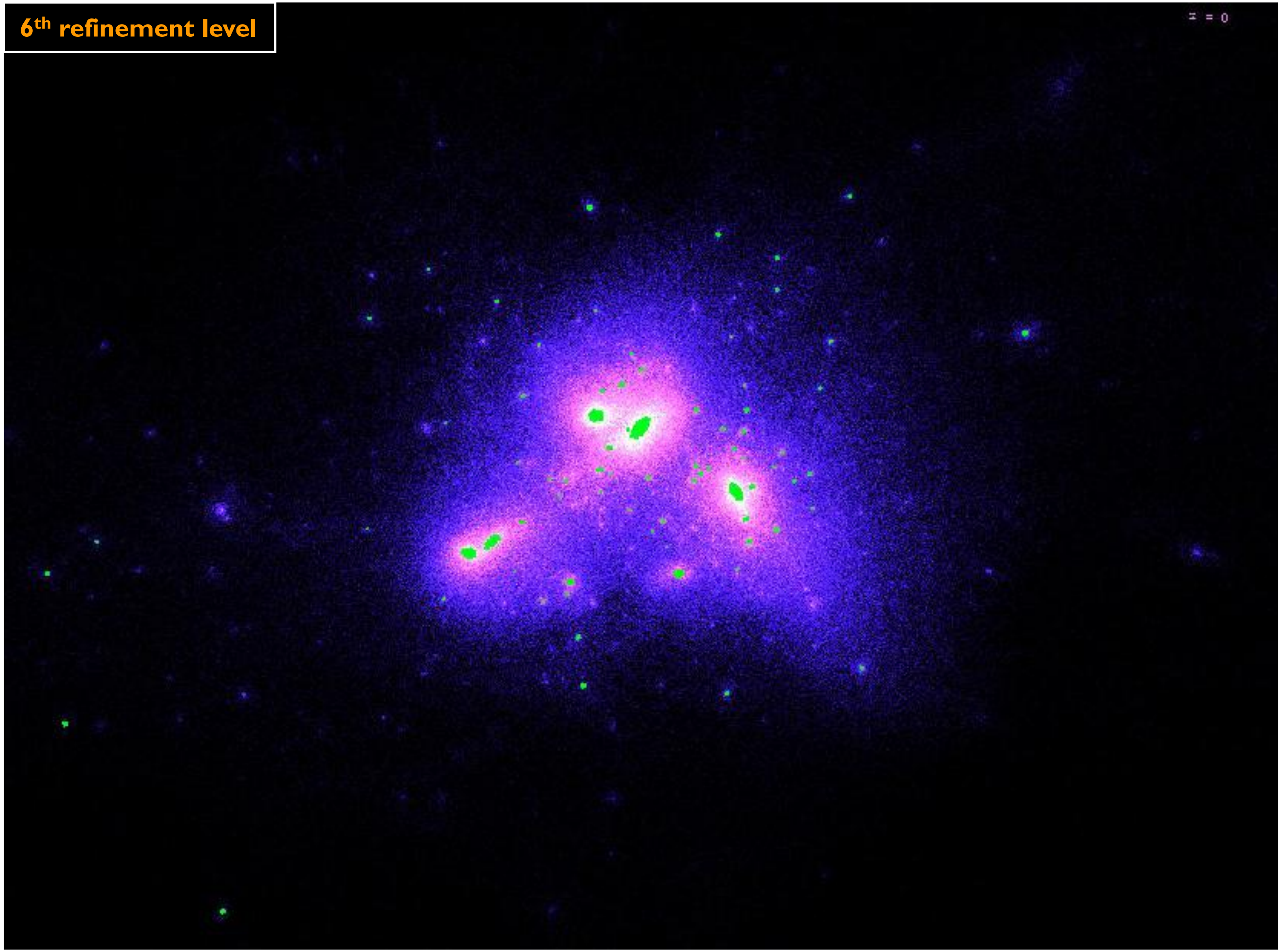
CONCEPT



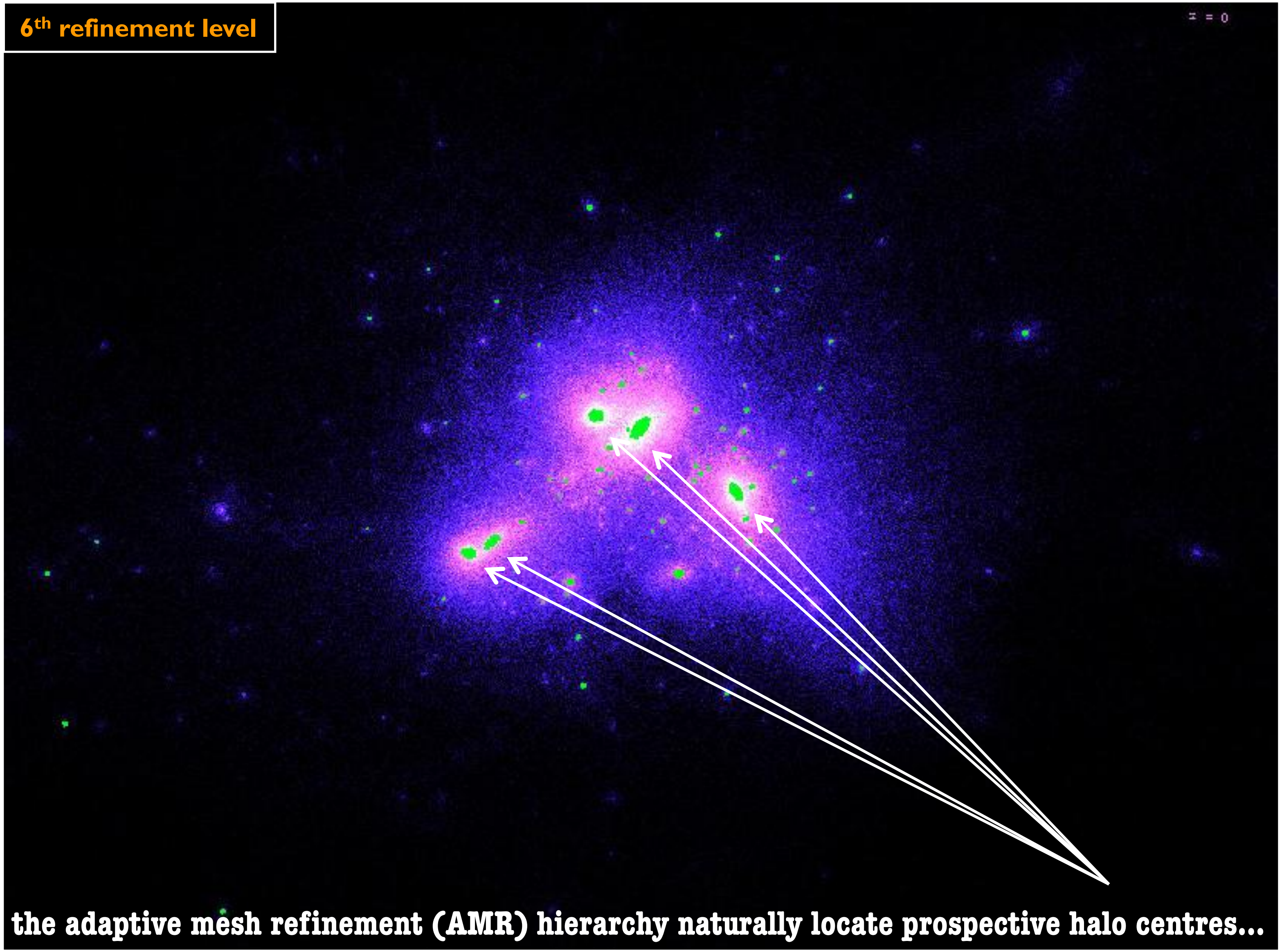
CONCEPT



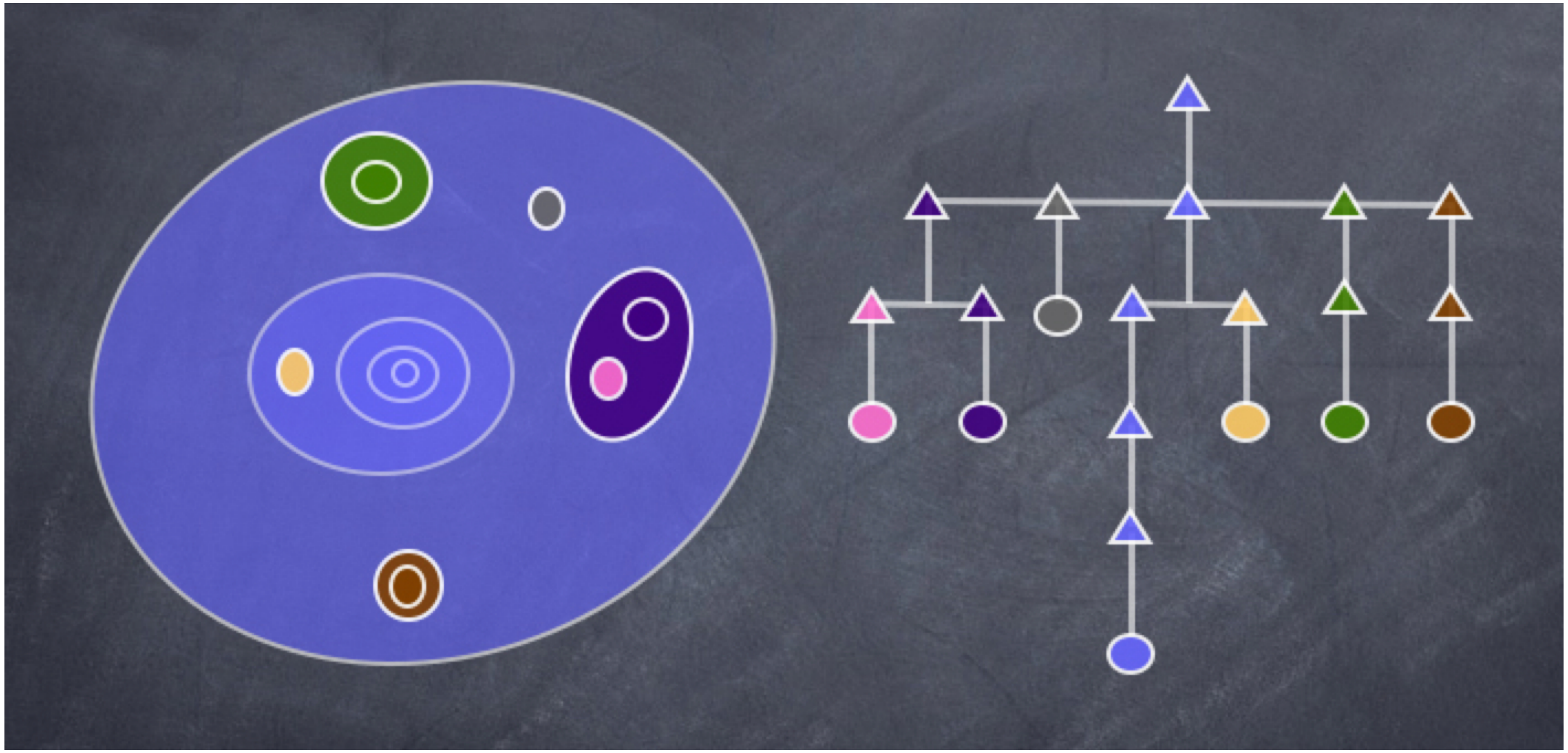
CONCEPT



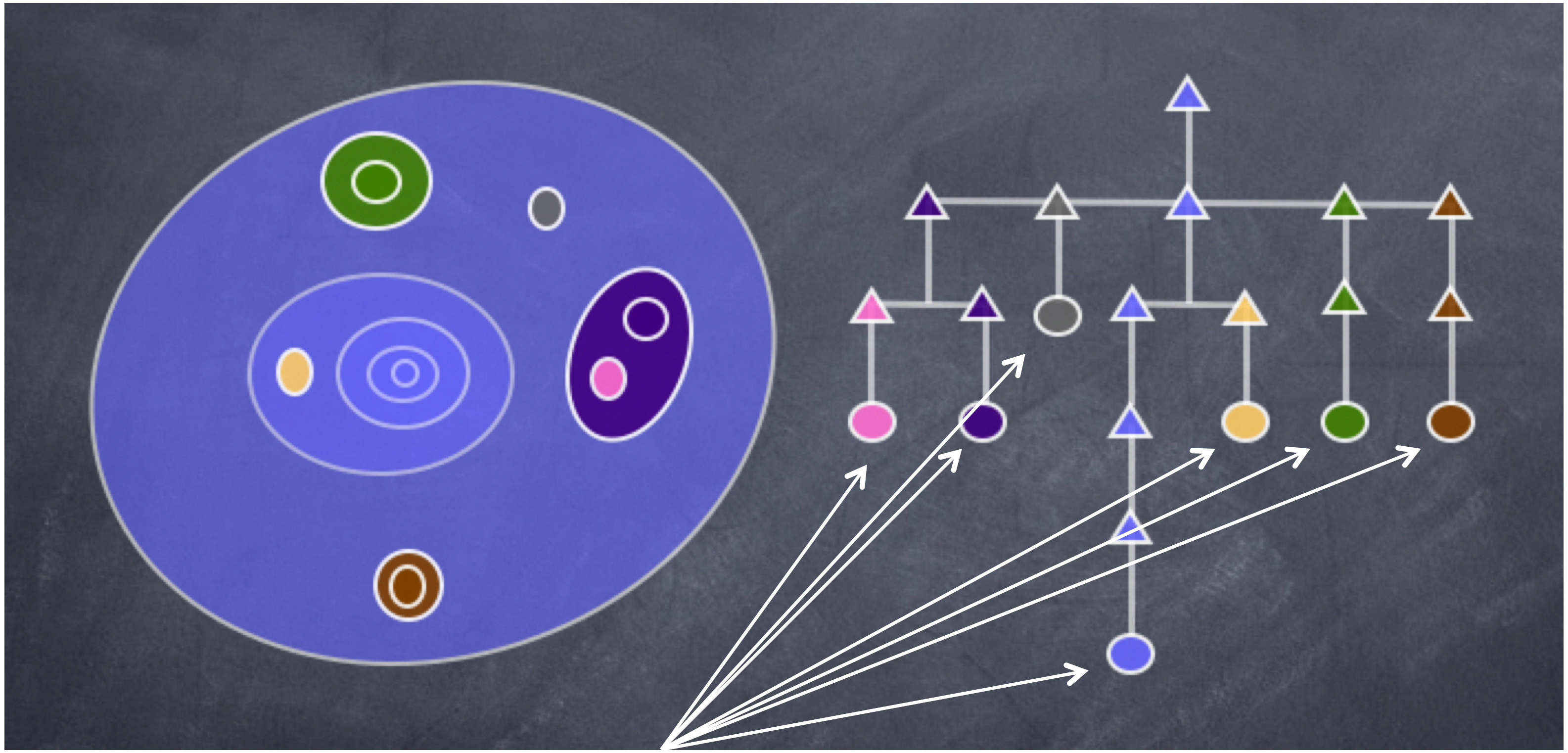
CONCEPT


6th refinement level
the adaptive mesh refinement (AMR) hierarchy naturally locate prospective halo centres...

- organize AMR hierarchy into a tree structure

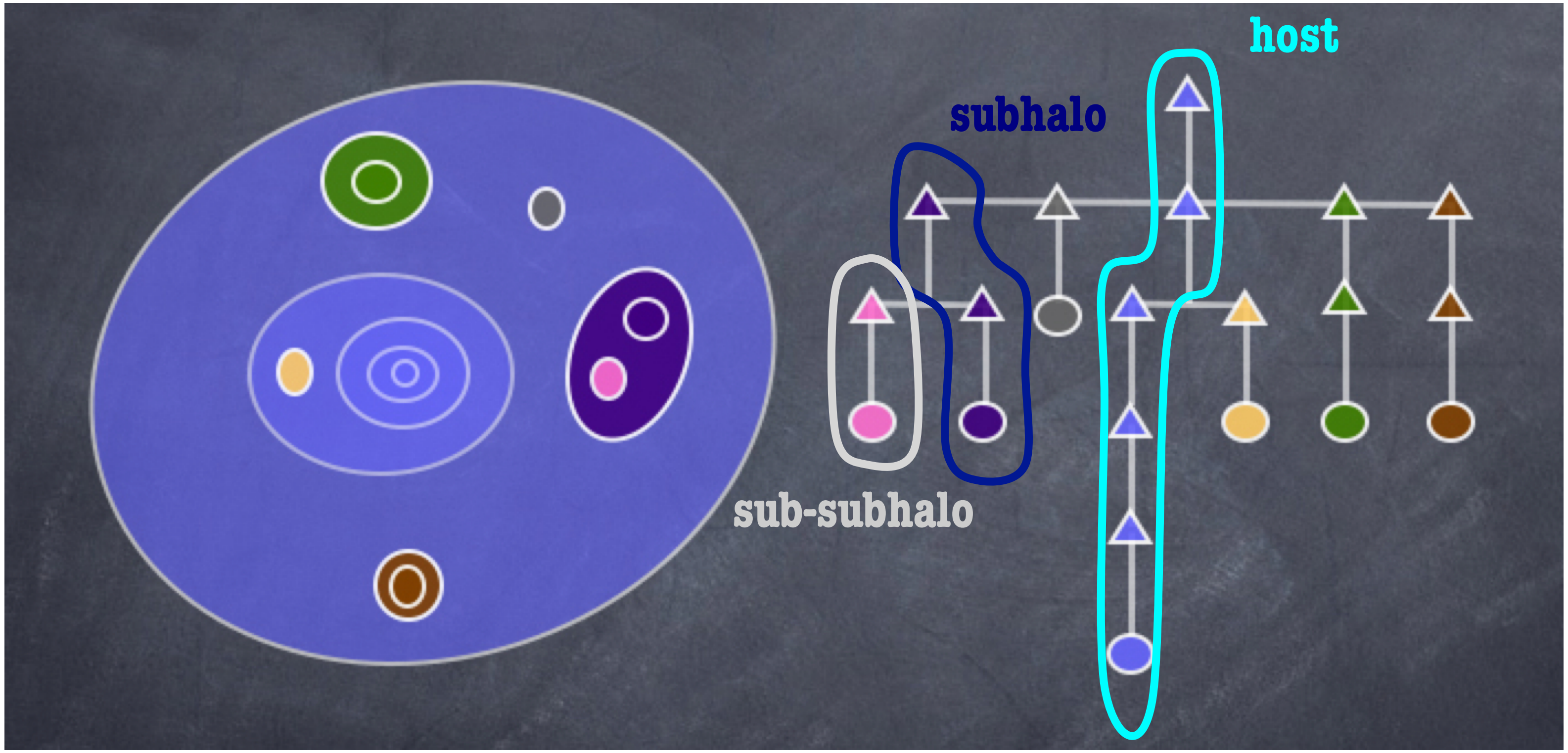


- organize AMR hierarchy into a tree structure



prospective halo centres...

- organize AMR hierarchy into a tree structure

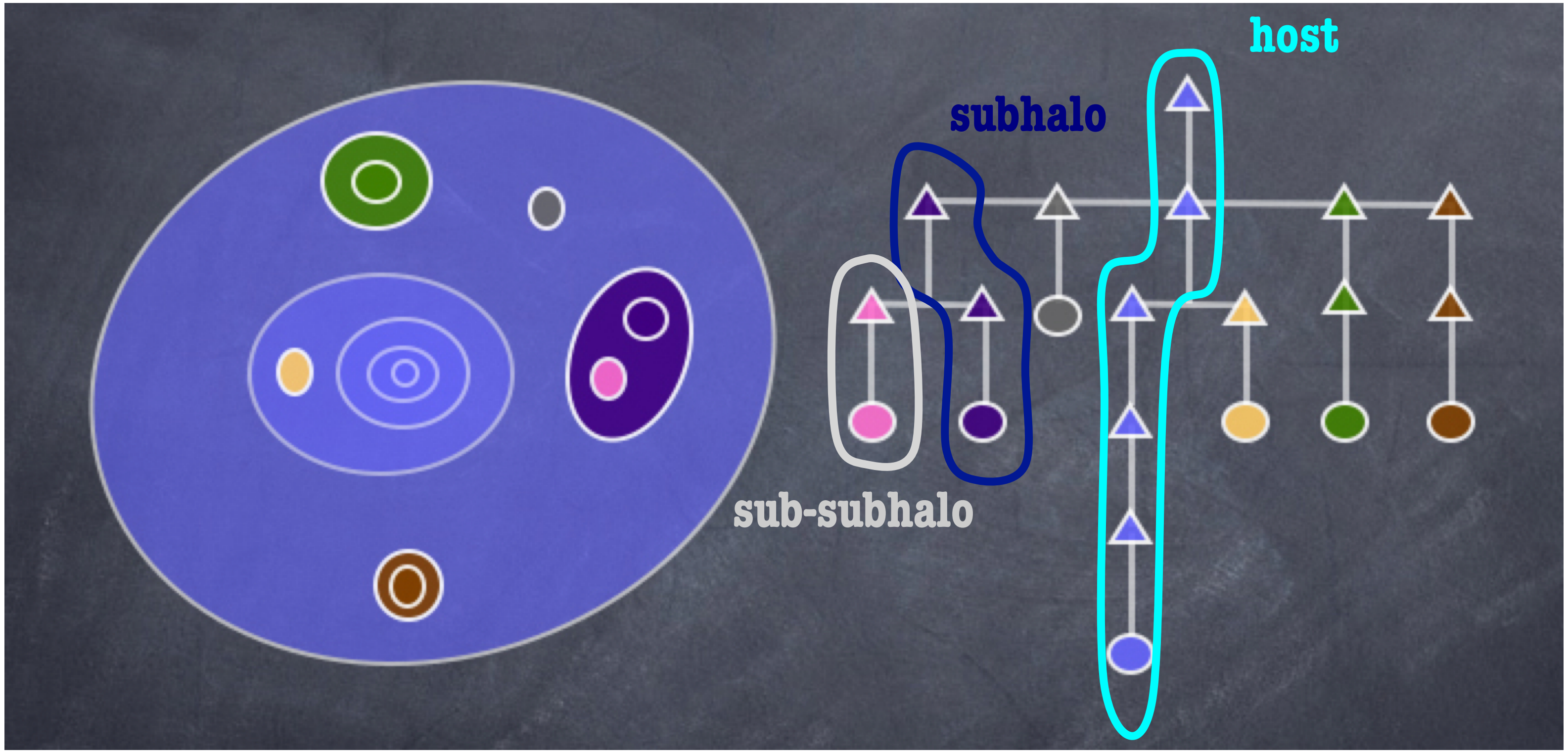


prospective halo centres...

...plus information about hosts, subhalos, sub-subhalos, etc.

- organize AMR hierarchy into a tree structure

The classification into host, subhalo, sub-subhalo, etc. will be explained later!

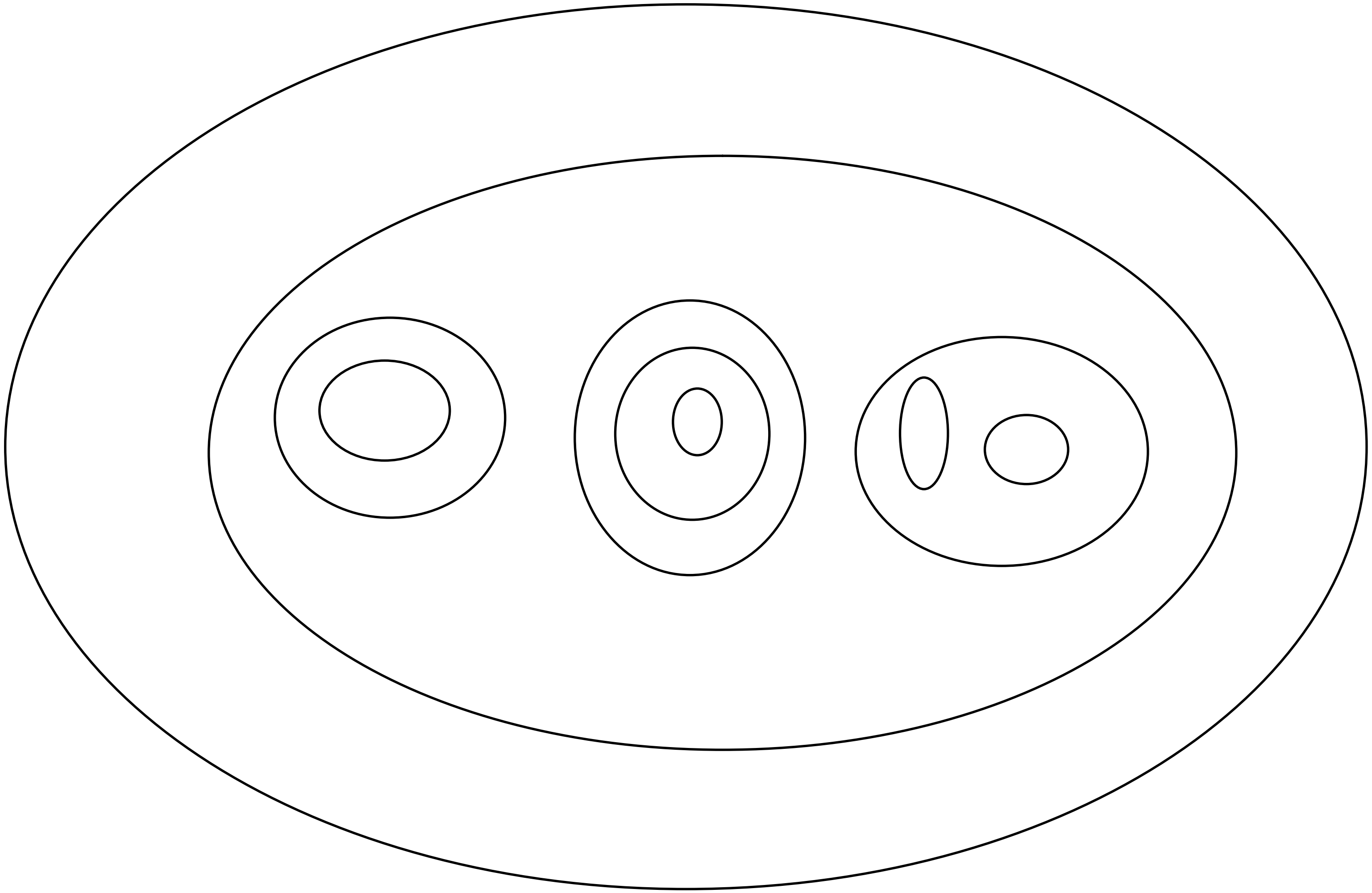


prospective halo centres...

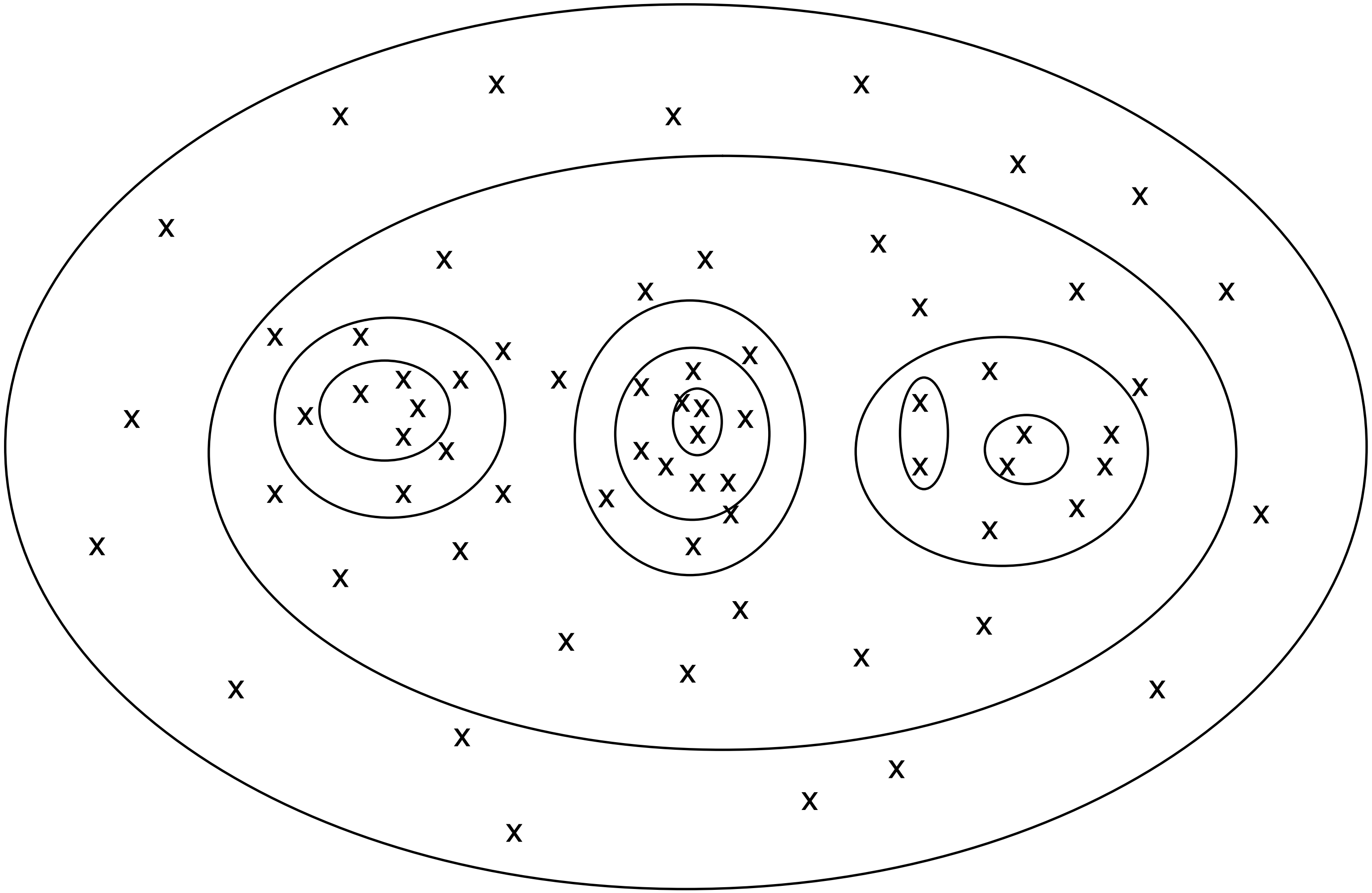
...plus information about hosts, subhalos, sub-subhalos, etc.

- AMR grids are isodensity contours

- AMR grids are isodensity contours

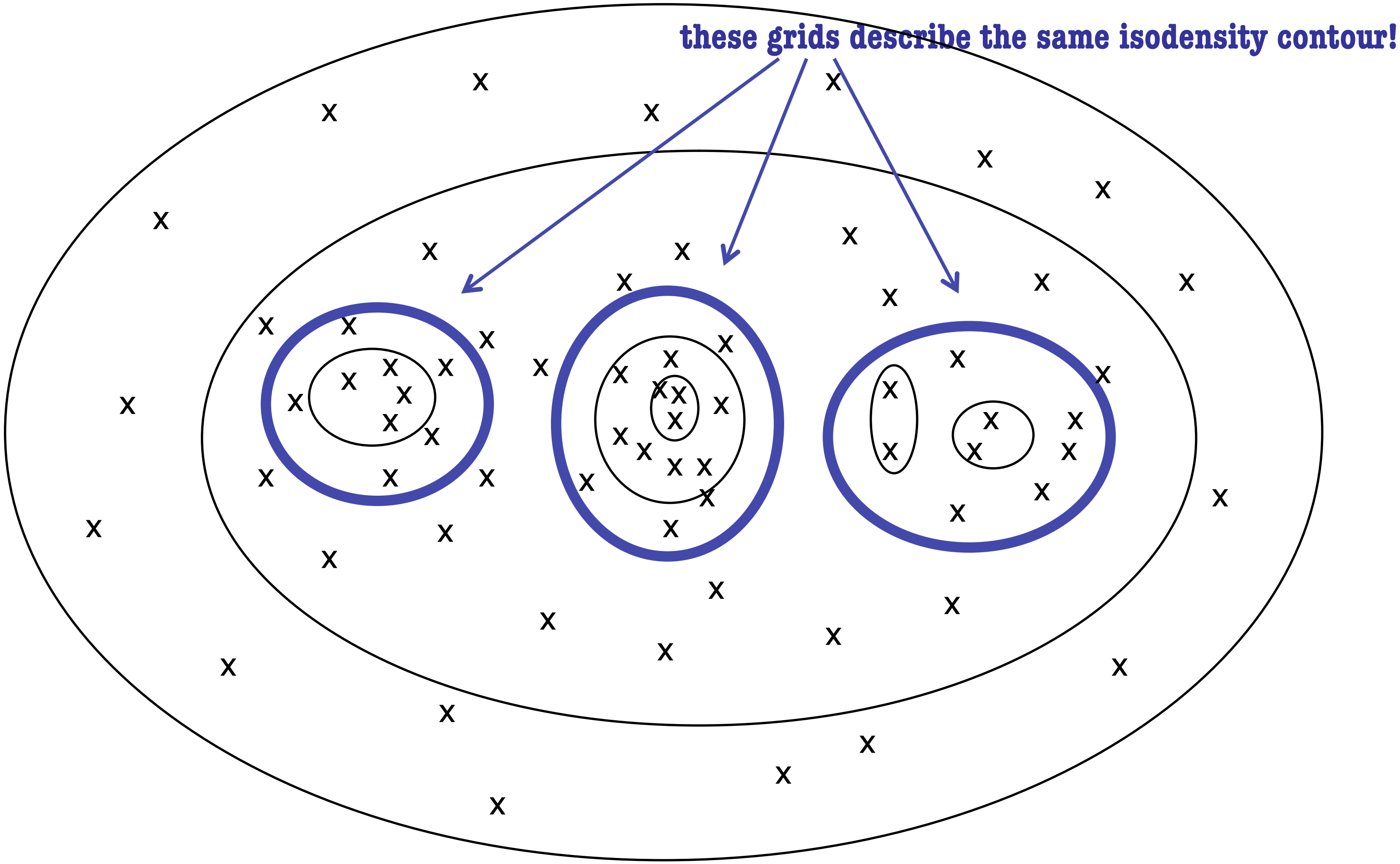


- AMR grids are isodensity contours...



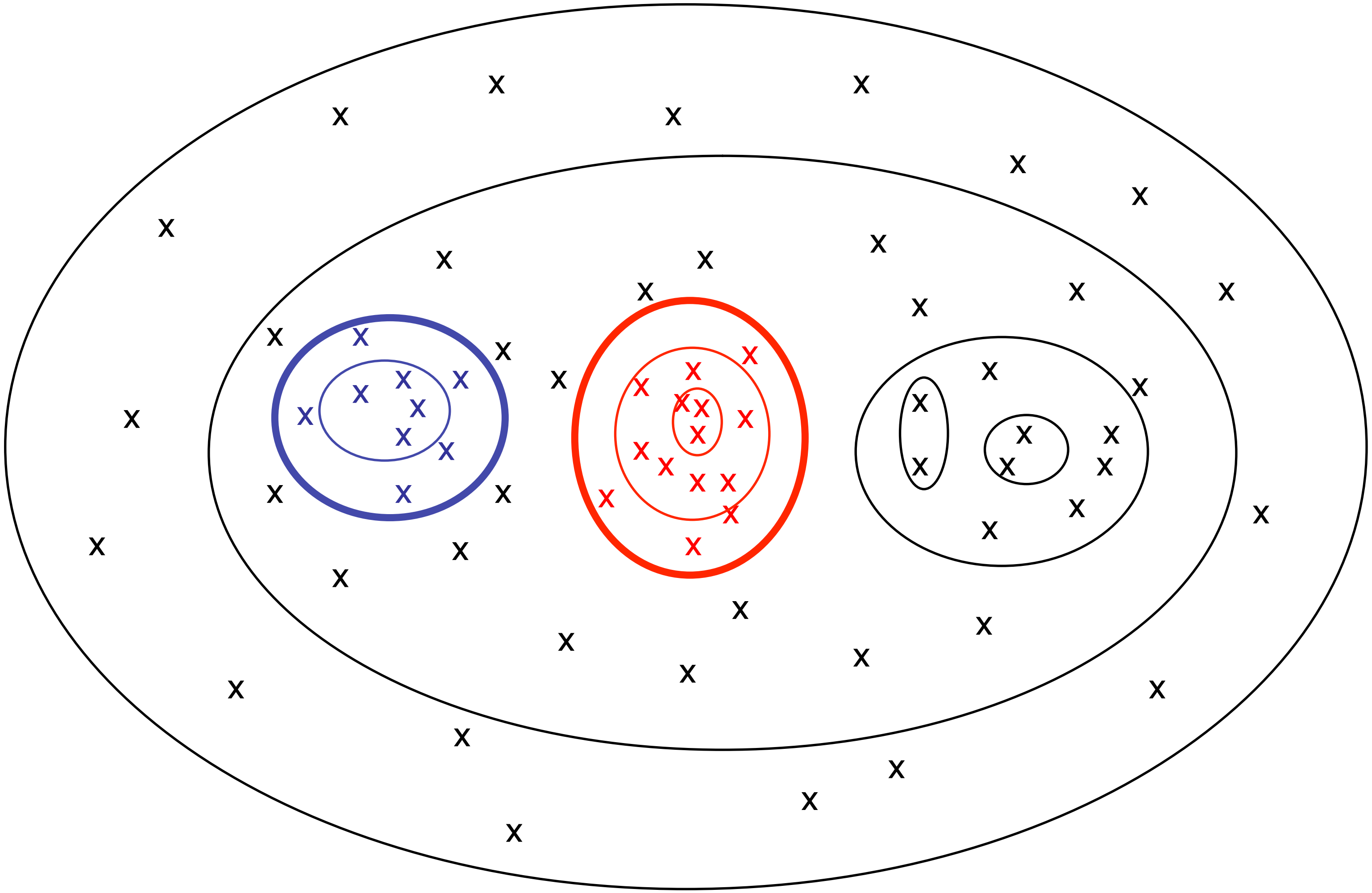
...encompassing particles!

- AMR grids are isodensity contours...

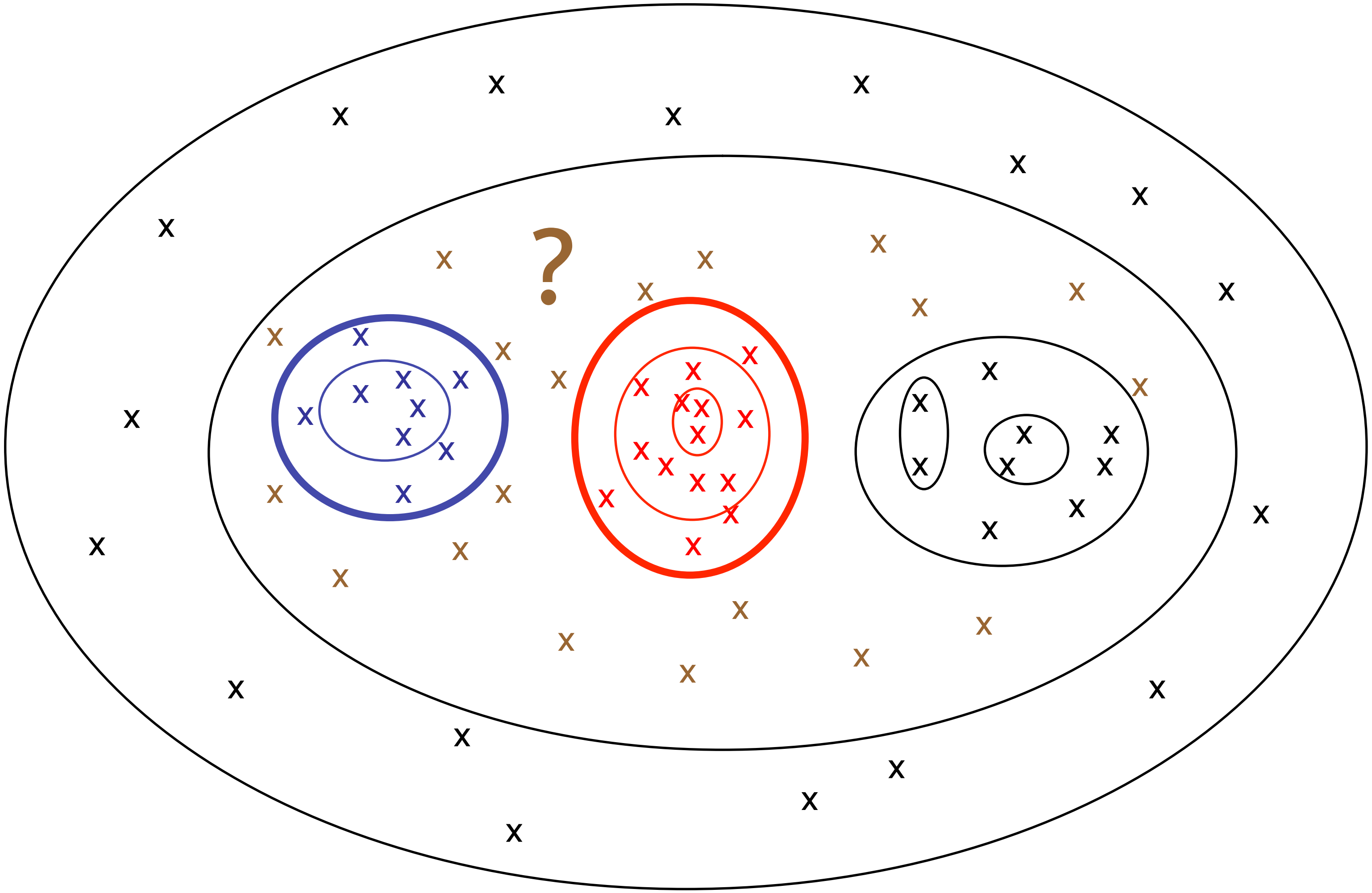


...encompassing particles!

1. collect all particles inside **unambiguous** isodensity contour

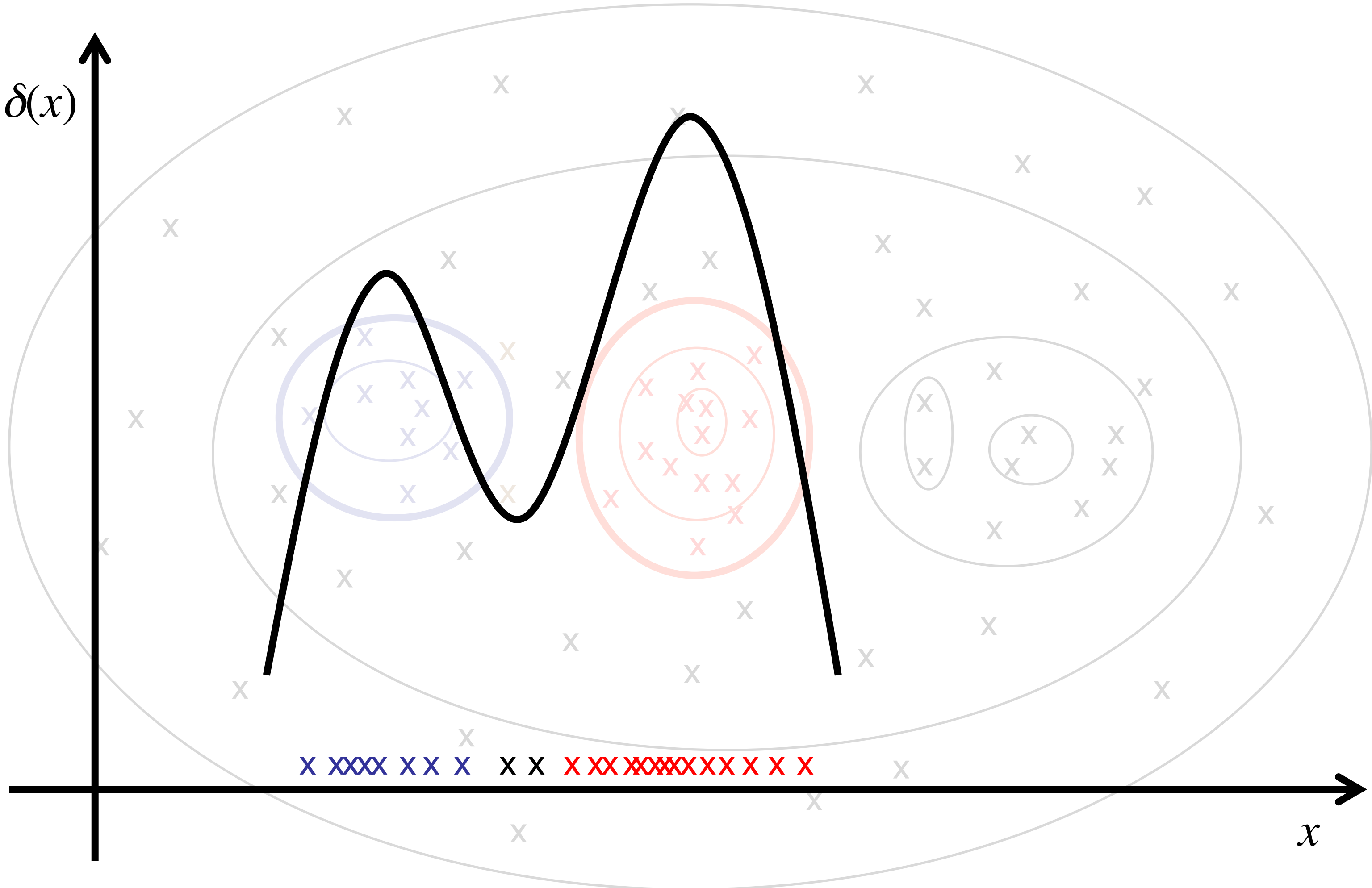


1. collect all particles inside **unambiguous** isodensity contour

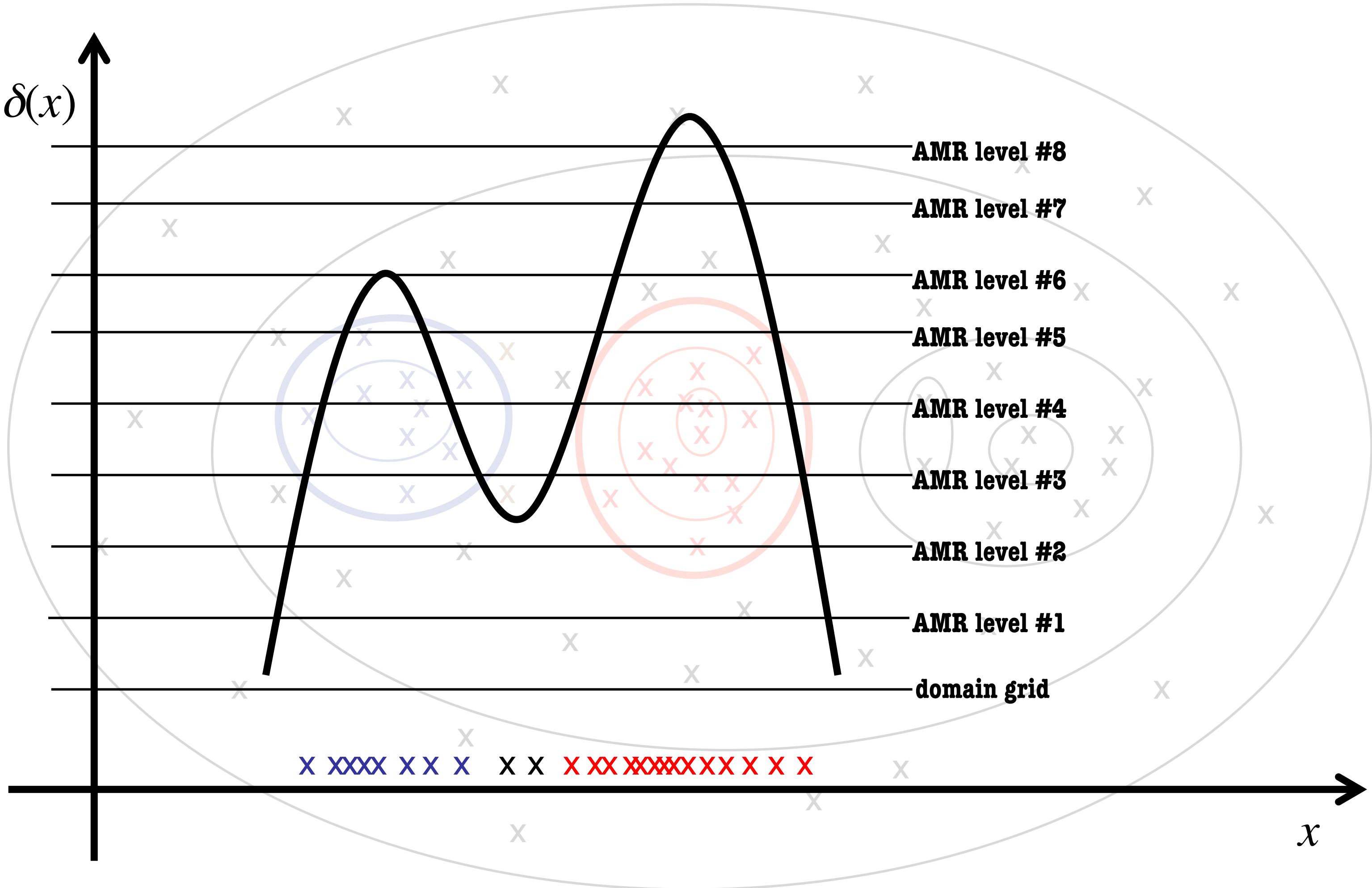


but what about **these** particles?

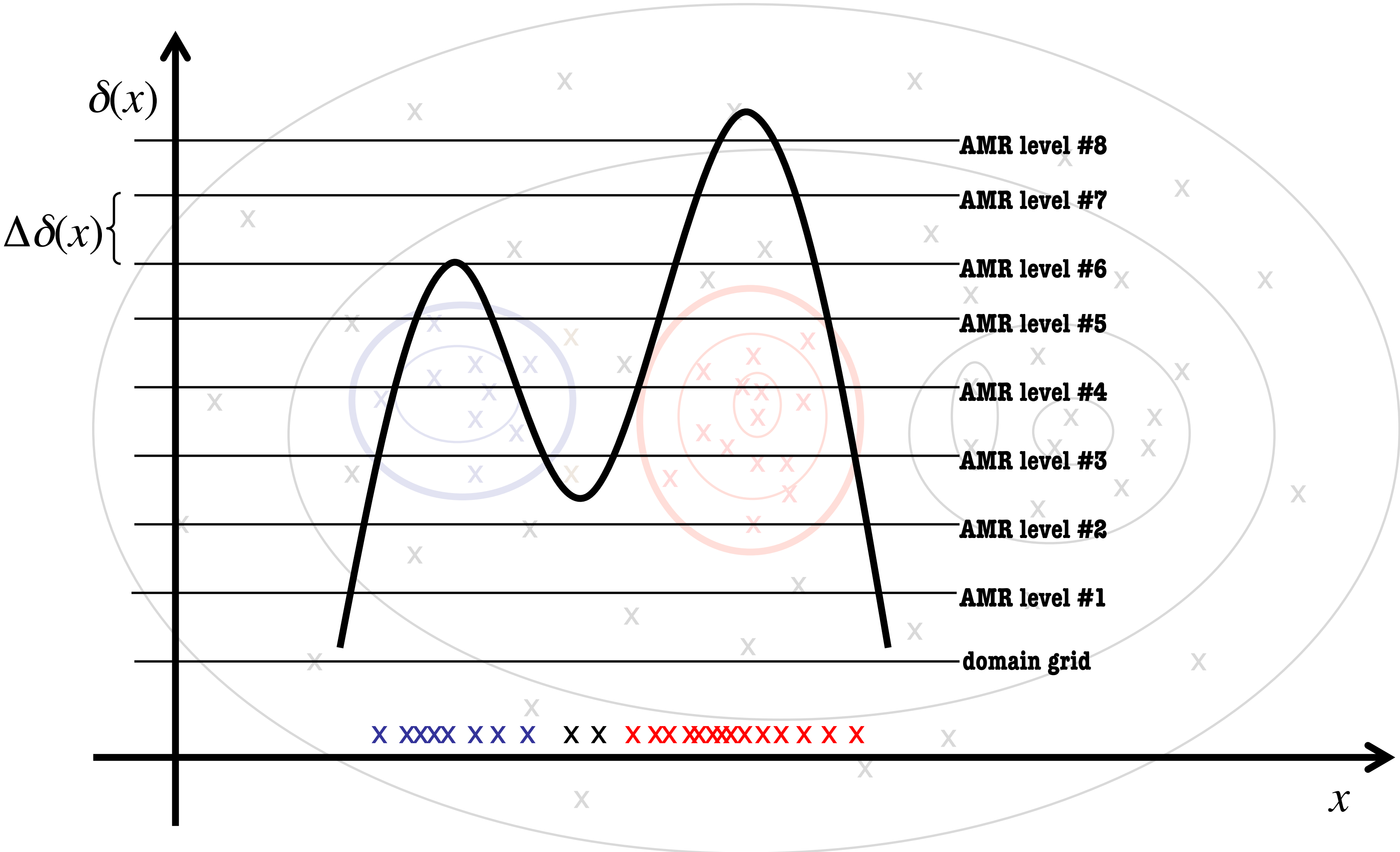
2. consider particles inside "half-distance-sphere", too!



2. consider particles inside "half-distance-sphere", too!

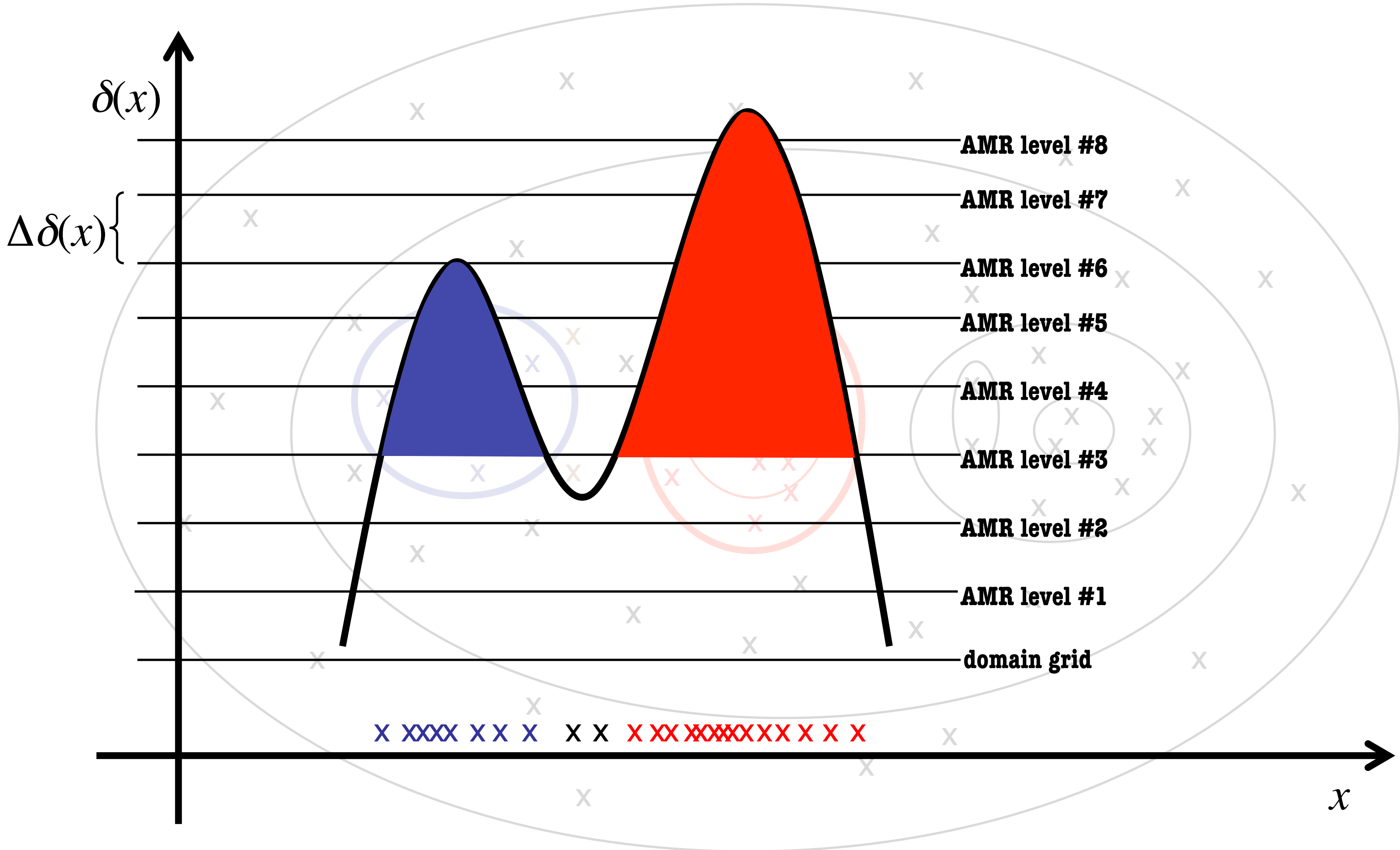


2. consider particles inside "half-distance-sphere", too!



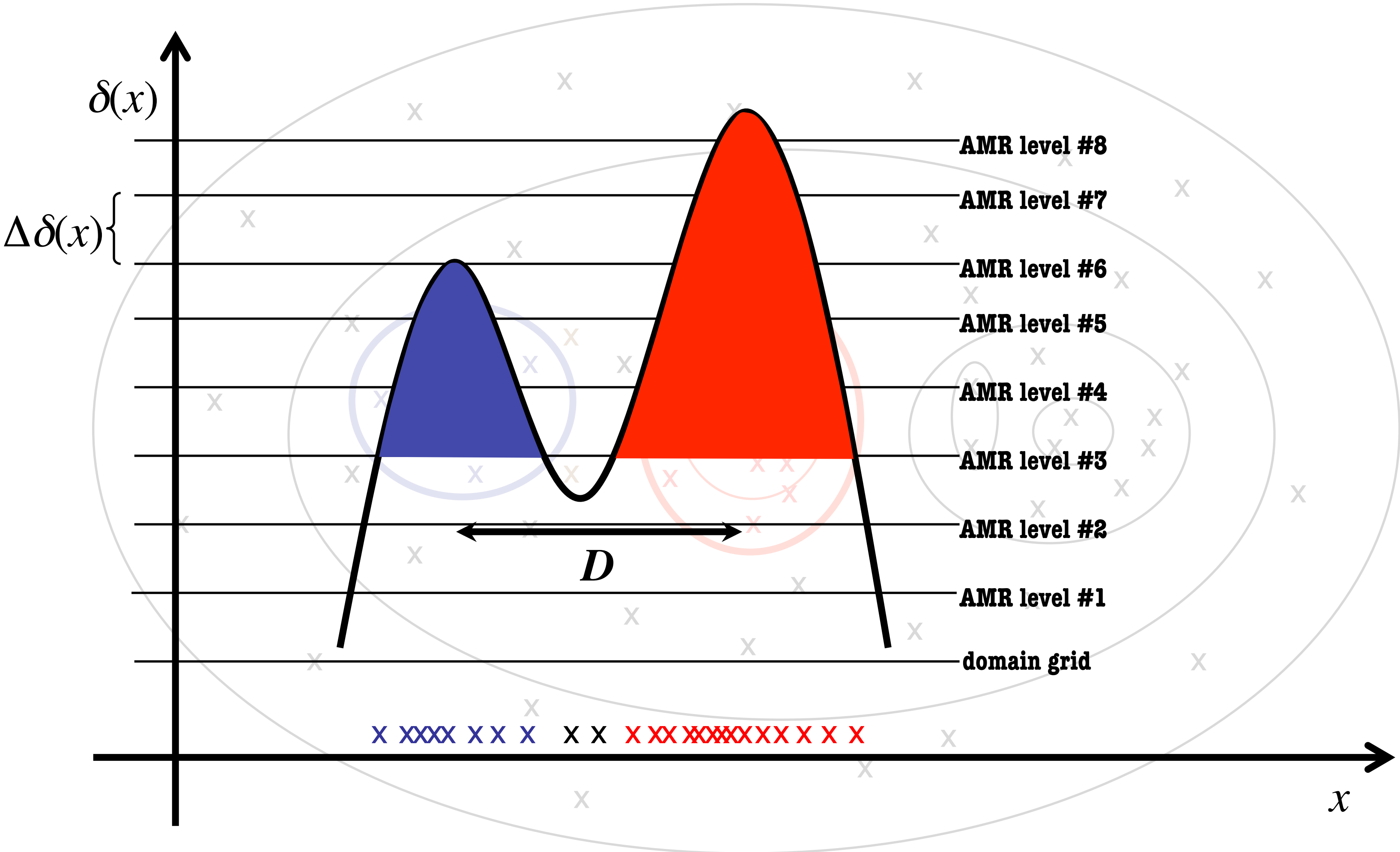
Note: $\Delta\delta(x)$ is determined by the refinement criterion...

2. consider particles inside "half-distance-sphere", too!



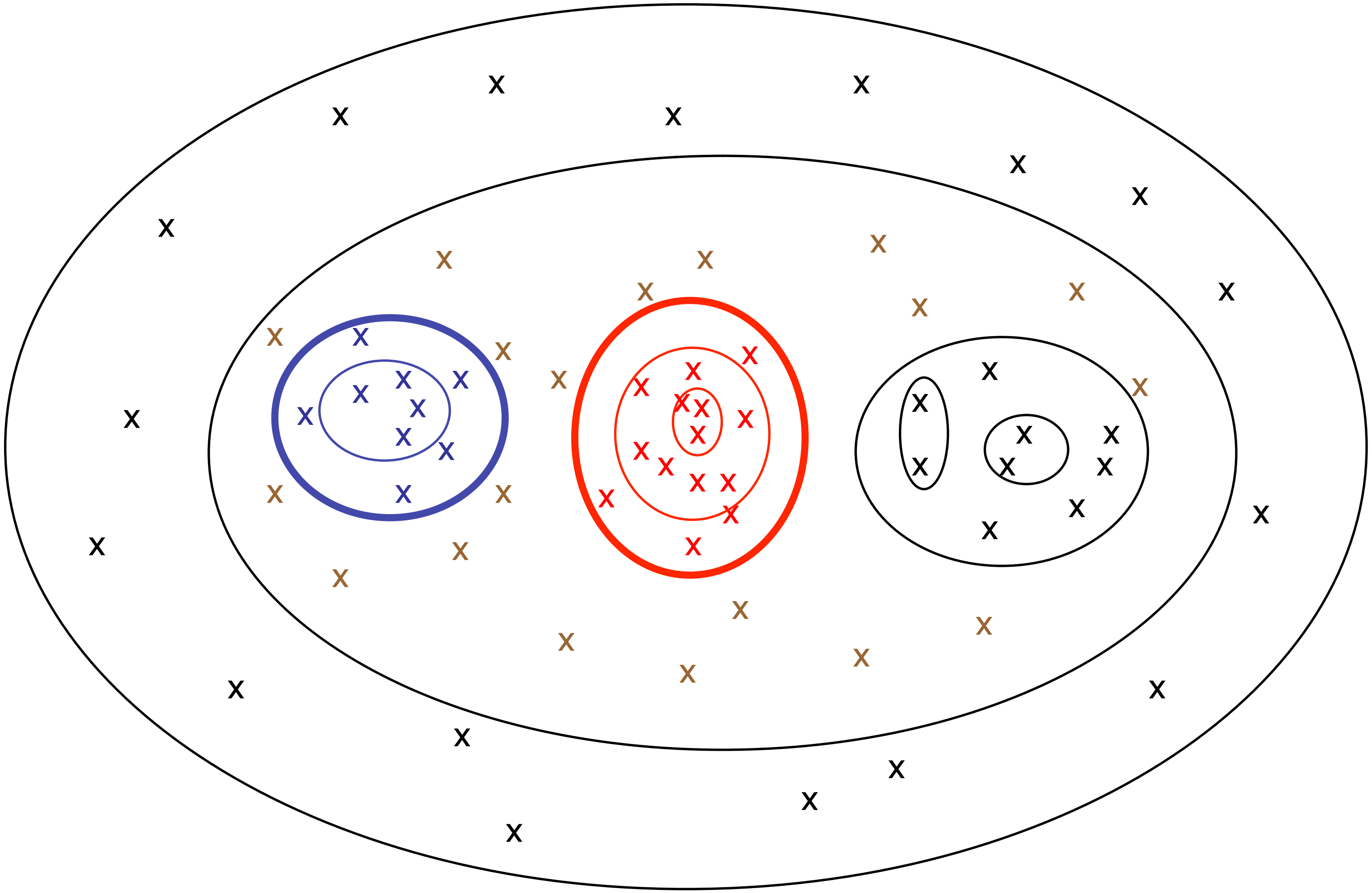
Note: $\Delta\delta(x)$ is determined by the refinement criterion...

2. consider particles inside "half-distance-sphere", too!

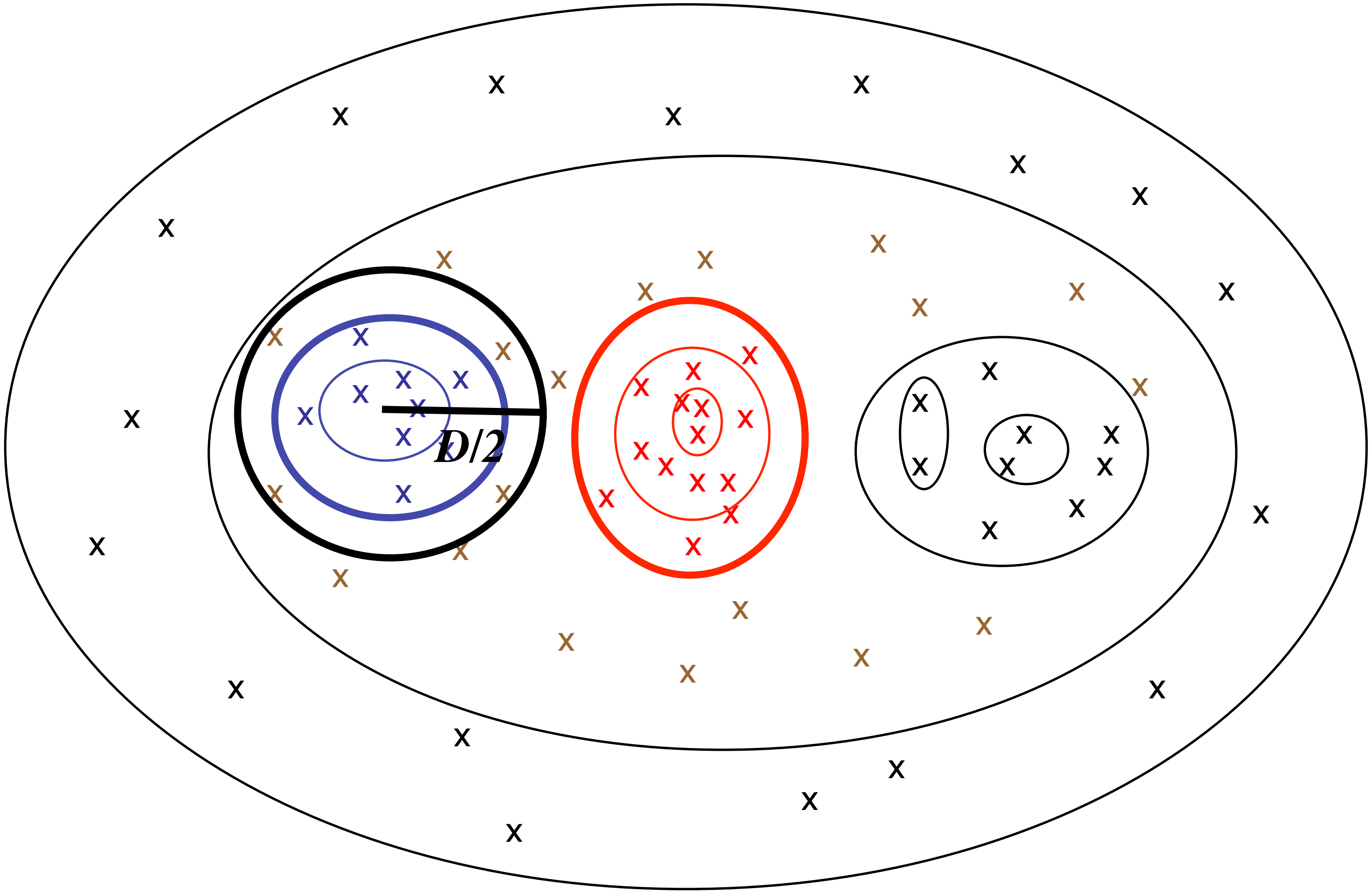


Note: $\Delta\delta(x)$ is determined by the refinement criterion...

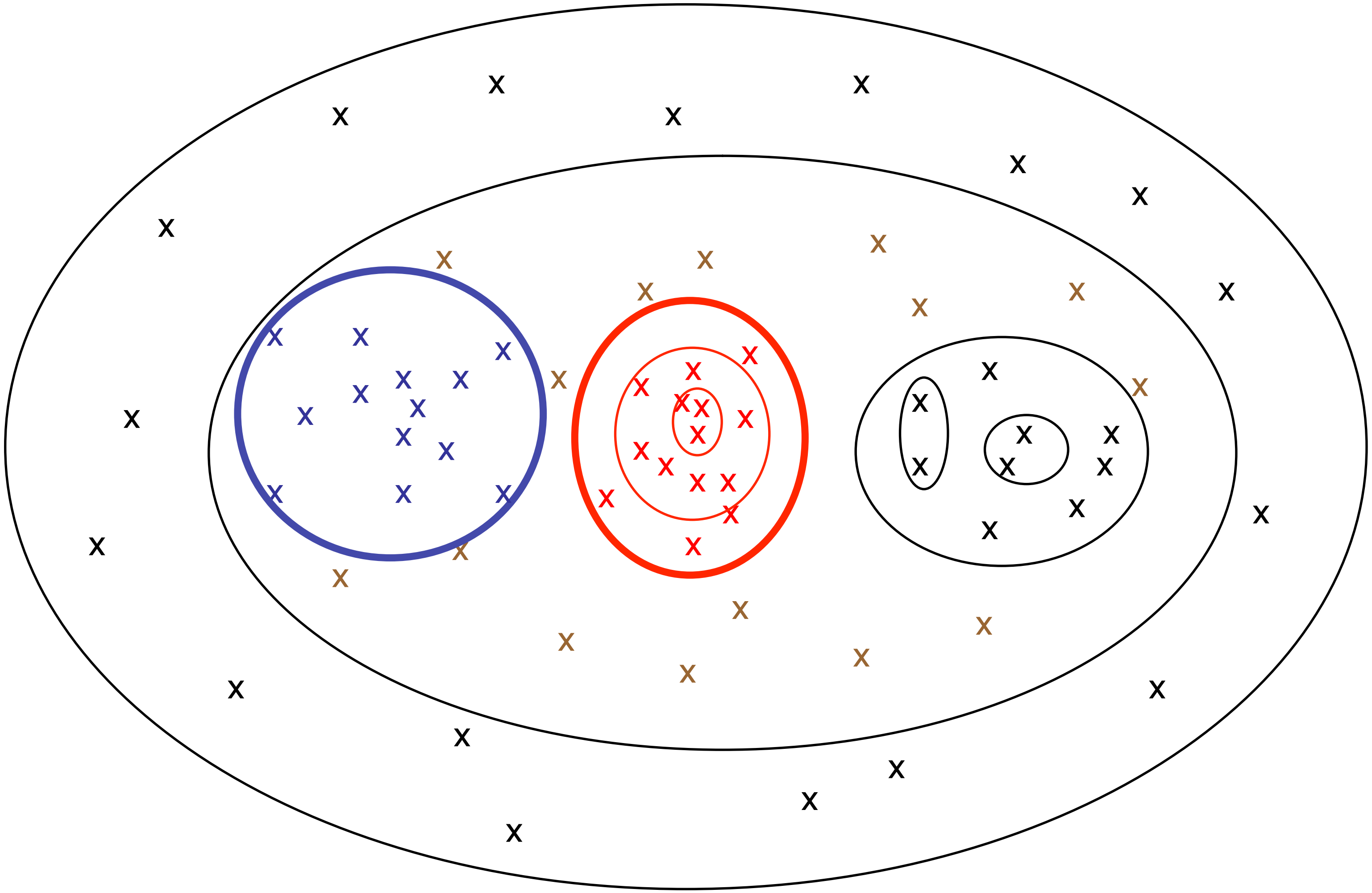
2. consider particles inside "half-distance-sphere", too!



2. consider particles inside "half-distance-sphere", too!

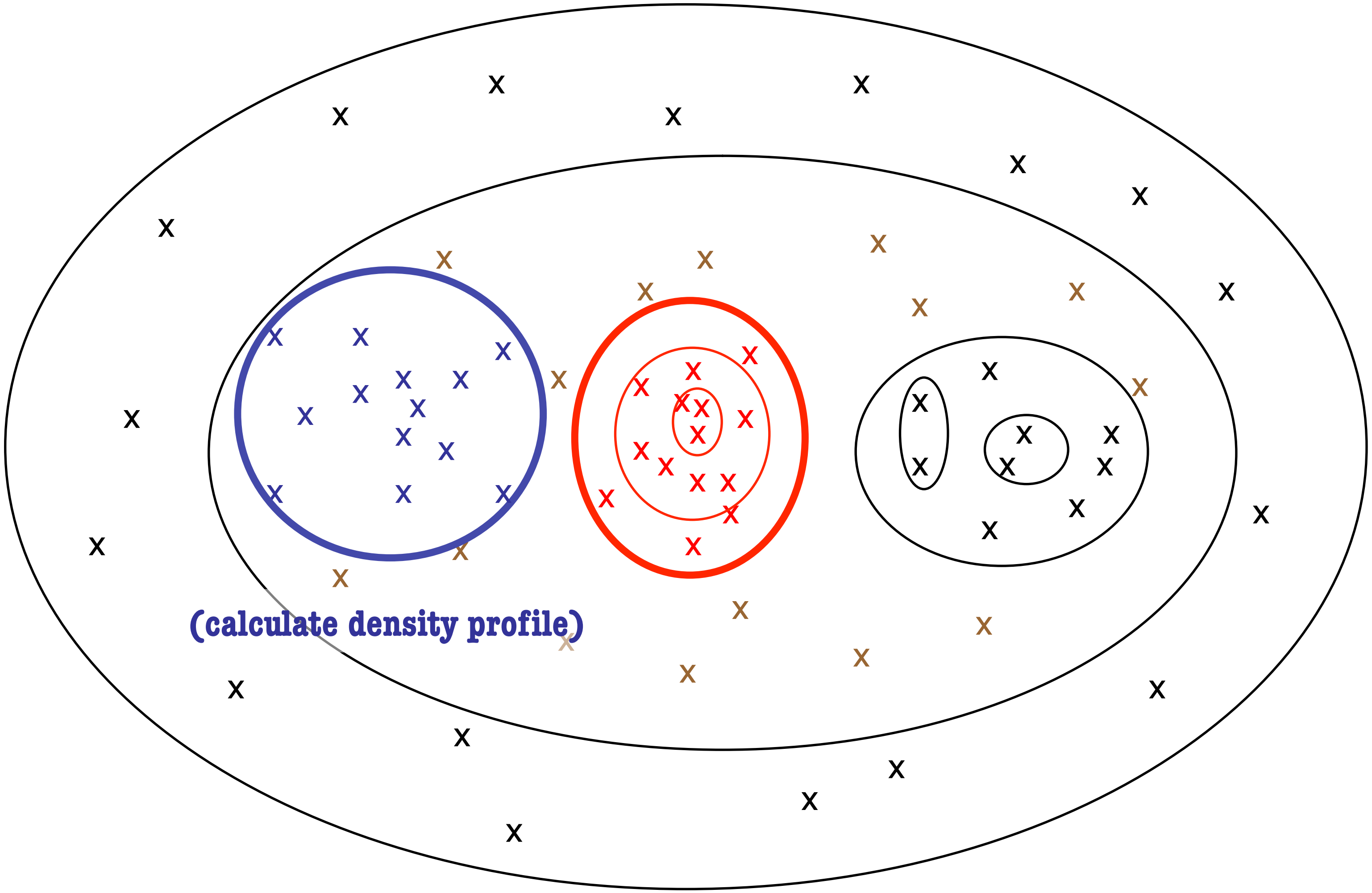


2. consider particles inside "half-distance-sphere", too!



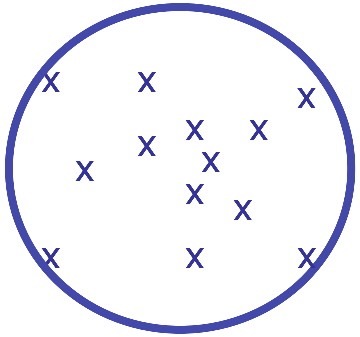
Note: particles not bound to the one **halo**, will be considered for boundness by the other **halo**

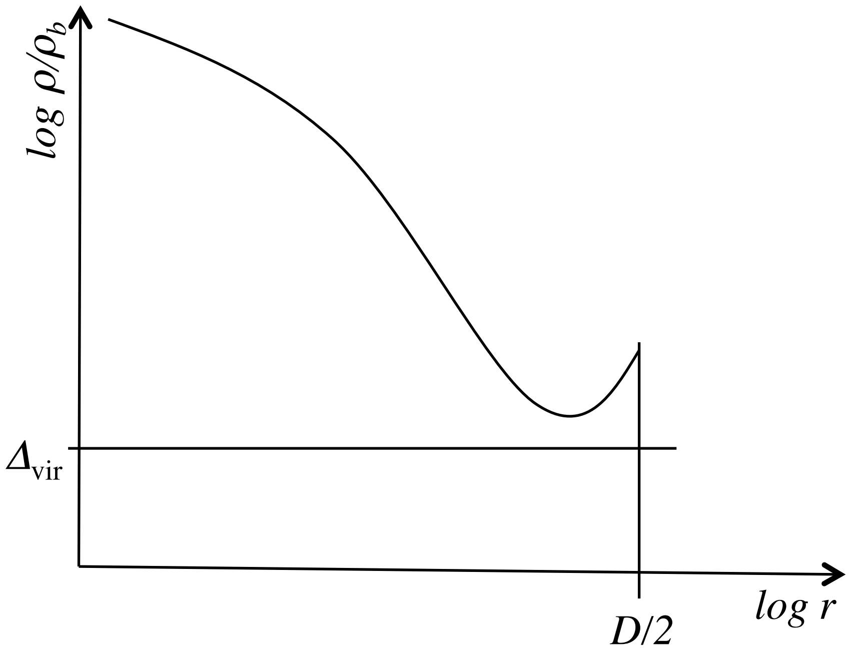
2. consider particles inside "half-distance-sphere", too!



which halo comes first relates to the classification into host, subhalo, sub-subhalo, etc...

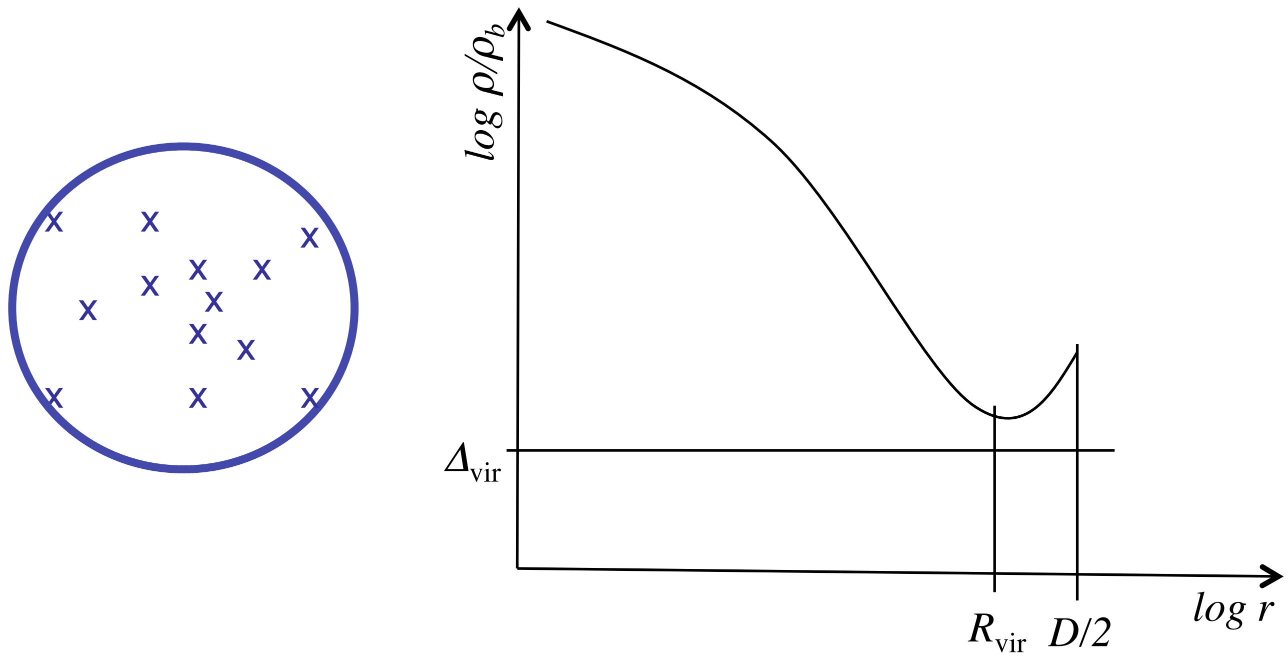
3. determine halo edge (prior to unbinding)





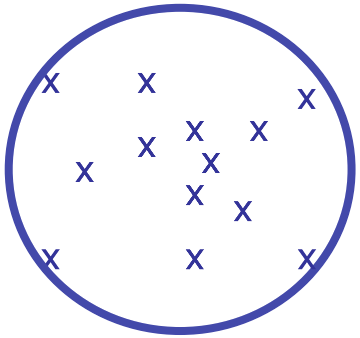
Δ_{vir} is calculated inside ***AHF*** and depends on cosmology and redshift!

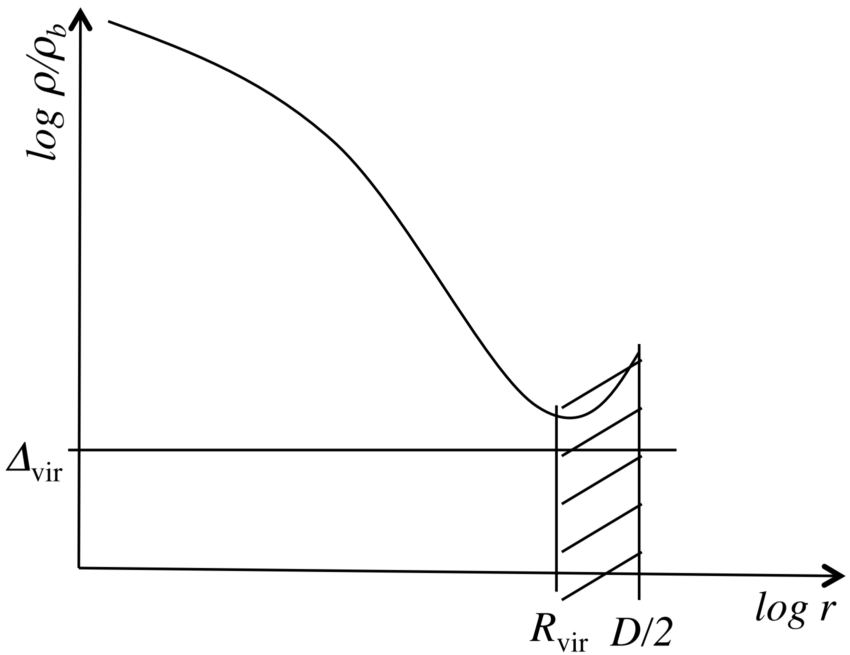
3. determine halo edge (prior to unbinding)



Δ_{vir} is calculated inside ***AHF*** and depends on cosmology and redshift!

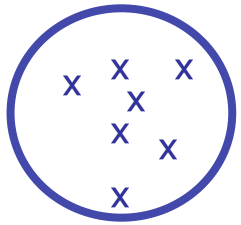
3. determine halo edge (prior to unbinding)

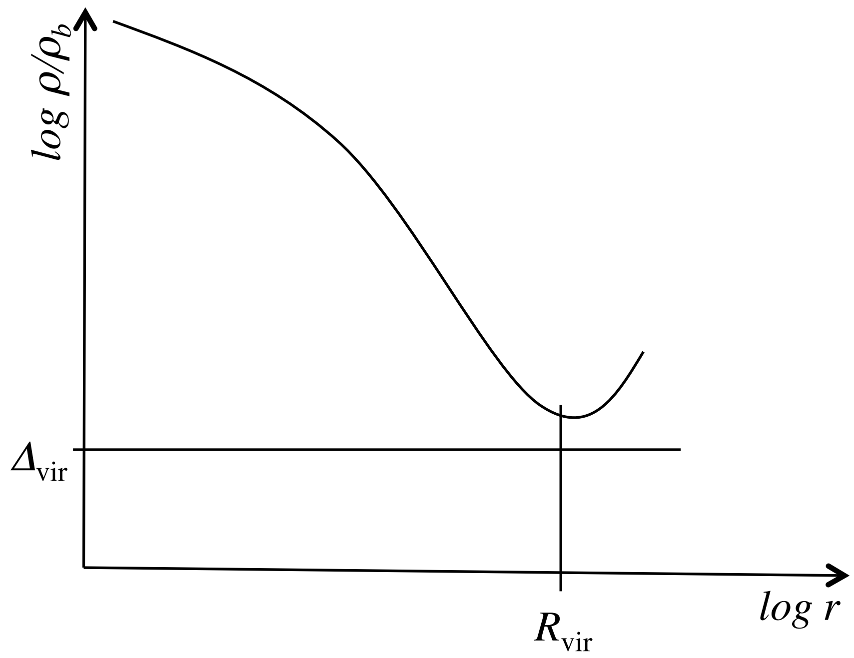




Δ_{vir} is calculated inside ***AHF*** and depends on cosmology and redshift!

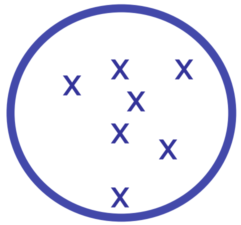
3. determine halo edge (prior to unbinding)





Δ_{vir} is calculated inside ***AHF*** and depends on cosmology and redshift!

4. iteratively remove unbound particles

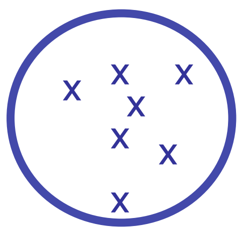


(remove unbound particles)

4. iteratively remove unbound particles

assume spherical symmetry:

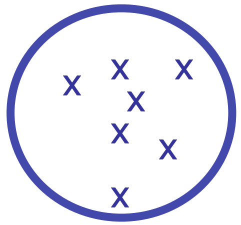
$$\rho = \rho(r)$$



(remove unbound particles)

AHF - AMIGA's HALO FINDER

4. iteratively remove unbound particles



(remove unbound particles)

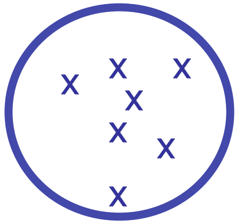
assume spherical symmetry:

$$\rho = \rho(r)$$

solve Poisson's equation:

$$\Delta\varphi = 4\pi G\rho$$

4. iteratively remove unbound particles



(remove unbound particles)

assume spherical symmetry:

$$\rho = \rho(r)$$

solve Poisson's equation:

$$\Delta\varphi = 4\pi G\rho$$

first integration...

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\varphi}{dr}\right) = 4\pi G\rho$$

$$\psi = r^2\frac{d\varphi}{dr}$$

$$\frac{1}{r^2}\frac{d}{dr}(\psi) = 4\pi G\rho$$

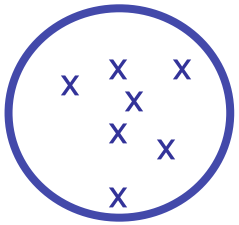
$$\frac{d\psi}{dr} = 4\pi G\rho r^2$$

$$\psi(r) - \psi(0) = 4\pi G\int_0^r \rho r'^2 dr'$$

$$\psi(0) = \left[r^2\frac{d\varphi}{dr}\right]_{r=0} = 0$$

$$\psi(r) = GM(<r)$$

4. iteratively remove unbound particles



(remove unbound particles)

assume spherical symmetry:

$$\rho = \rho(r)$$

solve Newton's force law:

$$\frac{d\varphi}{dr} = \frac{GM(<r)}{r^2}$$

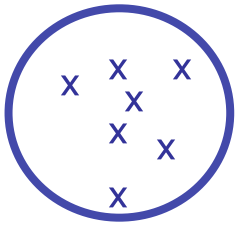
second integration...

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}\,dr' + \varphi(0)$$

unbound, if...

$$v > v_{\mathrm{esc}} = \sqrt{2|\varphi|}$$

4. iteratively remove unbound particles



(remove unbound particles)

assume spherical symmetry:

$$\rho = \rho(r)$$

solve Newton's force law:

$$\frac{d\varphi}{dr} = \frac{GM(<r)}{r^2}$$

second integration...

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}\,dr' + \varphi(0)\ ?$$

unbound, if...

$$v > v_{\text{esc}} = \sqrt{2|\varphi|}$$

4. iteratively remove unbound particles

assume spherical symmetry:

$$\rho = \rho(r)$$

solve Newton's force law:

$$\frac{d\varphi}{dr} = \frac{GM(<r)}{r^2}$$

potential normalisation:

$$\varphi(\infty) = G\int_0^\infty \frac{M(<r')}{r'^2}dr' + \varphi(0)$$
$$= G\int_0^{R_{vir}} \frac{M(<r')}{r'^2}dr' + G\int_{R_{vir}}^\infty \frac{M(<r')}{r'^2}dr' + \varphi(0)$$
$$= G\int_0^{R_{vir}} \frac{M(<r')}{r'^2}dr' + GM_{vir}\int_{R_{vir}}^\infty \frac{1}{r'^2}dr' + \varphi(0)$$
$$= G\int_0^{R_{vir}} \frac{M(<r')}{r'^2}dr' + GM_{vir}\left[-\frac{1}{r}\right]_{R_{vir}}^\infty + \varphi(0)$$
$$= G\int_0^{R_{vir}} \frac{M(<r')}{r'^2}dr' + G\frac{M_{vir}}{R_{vir}} + \varphi(0)$$

second integration...

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}dr' + \varphi(0) \; ?$$

unbound, if...

$$v > v_{esc} = \sqrt{2|\varphi|}$$

4. iteratively remove unbound particles

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}\,dr' - \varphi_0$$

with:

$$\varphi_0 = G\left(\frac{M_{vir}}{R_{vir}} + \int_0^{R_{vir}} \frac{M(<r')}{r'^2}\,dr'\right)$$

the integrals can be readily evaluated in cosmological simulations...

4. iteratively remove unbound particles

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}dr' - \varphi_0$$

order particles with respect to distance:

$$\int_0^r \frac{M(<r')}{r'^2}dr' = \int_0^{r_1} \frac{M(<r)}{r^2}dr + \int_{r_1}^{r_2} \frac{M(<r)}{r^2}dr + \ldots + \int_{r_{N-1}}^{r_N} \frac{M(<r)}{r^2}dr$$

$$= \frac{m_1}{r_1^2}r_1 + \frac{m_1 + m_2}{r_2^2}|r_2 - r_1| + \frac{m_1 + m_2 + m_3}{r_3^2}|r_3 - r_2| + \ldots$$

4. **iteratively** remove unbound particles

$$\varphi(r) = G\int_0^r \frac{M(<r')}{r'^2}dr' - \varphi_0$$

order particles with respect to distance:

$$\int_0^r \frac{M(<r')}{r'^2}dr' = \int_0^{r_1} \frac{M(<r)}{r^2}dr + \int_{r_1}^{r_2} \frac{M(<r)}{r^2}dr + \ldots + \int_{r_{N-1}}^{r_N} \frac{M(<r)}{r^2}dr$$

$$= \frac{m_1}{r_1^2}r_1 + \frac{m_1+m_2}{r_2^2}|r_2 - r_1| + \frac{m_1+m_2+m_3}{r_3^2}|r_3 - r_2| + \ldots$$

AHF - AMIGA's HALO FINDER

4. iteratively remove unbound particles

1. obtain initial set of particles and determine $M_{\rm vir}$ and $R_{\rm vir}$

2. calculate $\varphi_0 = G\left(\frac{M_{vir}}{R_{vir}} + \int_0^{R_{vir}} \frac{M(<r')}{r'^2} dr'\right)$

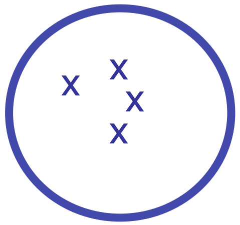
3. while looping over all particles check $v_i > v_{\rm esc}(r_i) = \sqrt{2|\varphi(r_i)|}$

4. using $\varphi(r_i) = G\int_0^{r_i} \frac{M(<r)}{r^2} dr - \varphi_0$

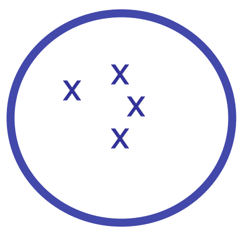
5. bound particles define a new set of initial particles for $M_{\rm vir}$ and $R_{\rm vir}$

$\Rightarrow$ start from 2. again and repeat until no further unbound particles...

4. iteratively remove unbound particles

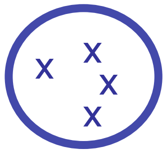


5. determine halo edge (final)



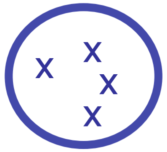
(re-adjust radius)

5. determine halo edge (final)



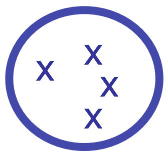
6. eventually determine halo properties

R_{vir} = { the point where the density profile of bound particles drops below $\Delta_{\mathrm{vir}}\rho_b$; distance to farthest bound particle within "tidal radius" }



6. eventually determine halo properties

$$R_{\rm vir} = \begin{cases} \text{the point where the density profile of bound particles drops below } \Delta_{\rm vir}\rho_b \\ \text{distance to farthest bound particle within ``tidal radius''} \end{cases}$$

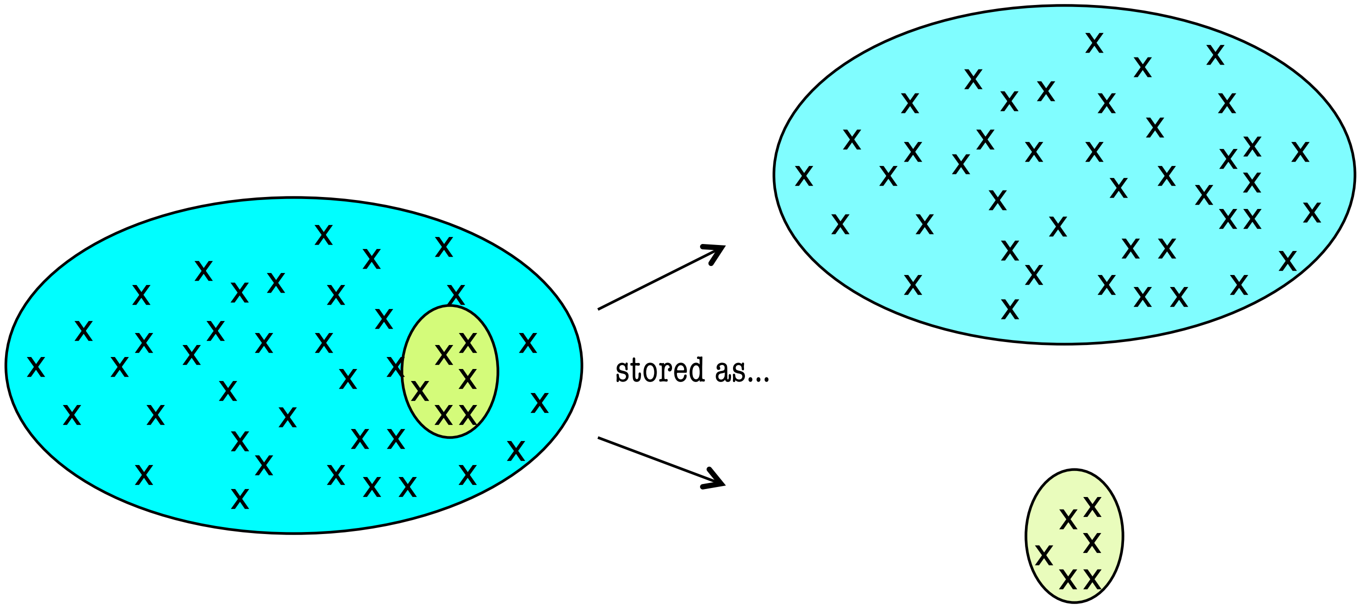


all other halo properties are based upon particles

inside sphere of radius $R_{\rm vir}$!

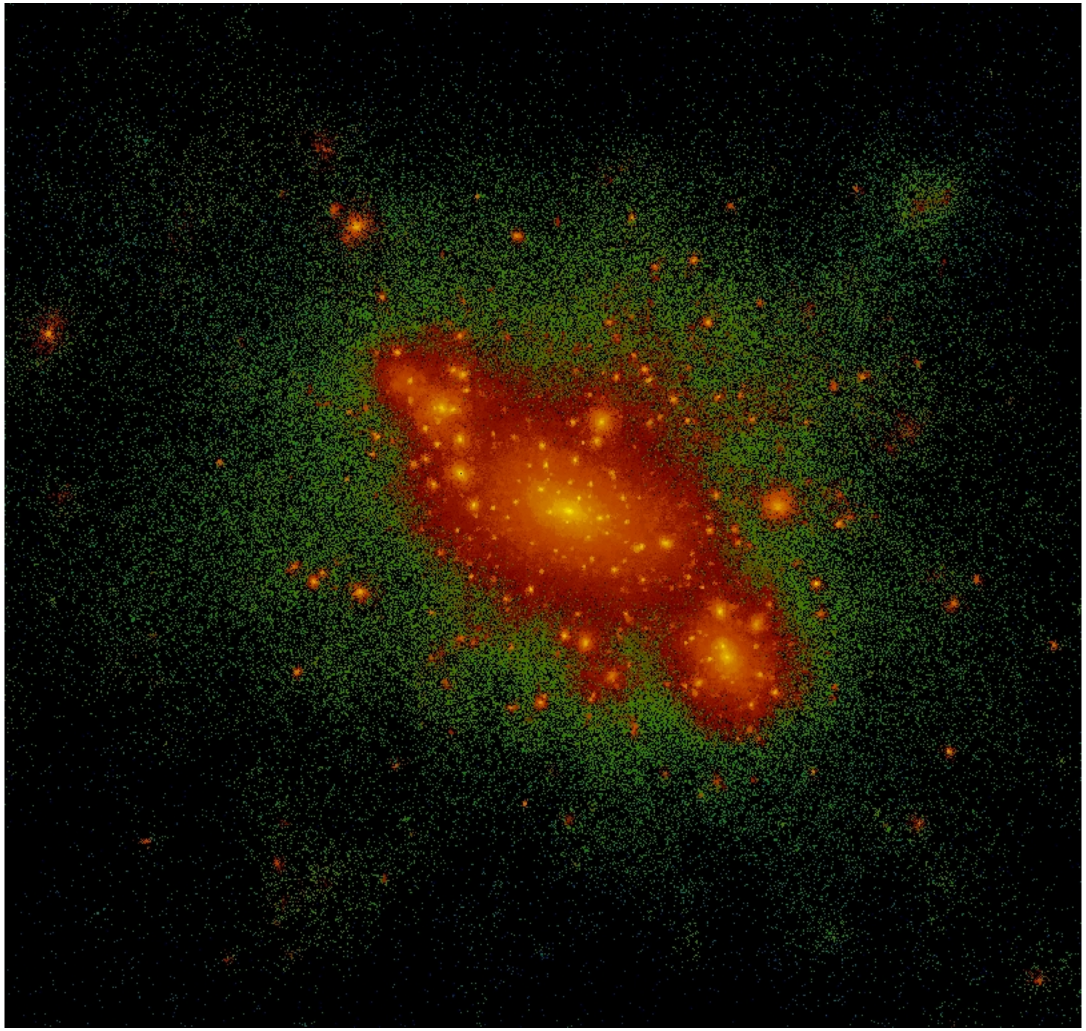
6. eventually determine halo properties

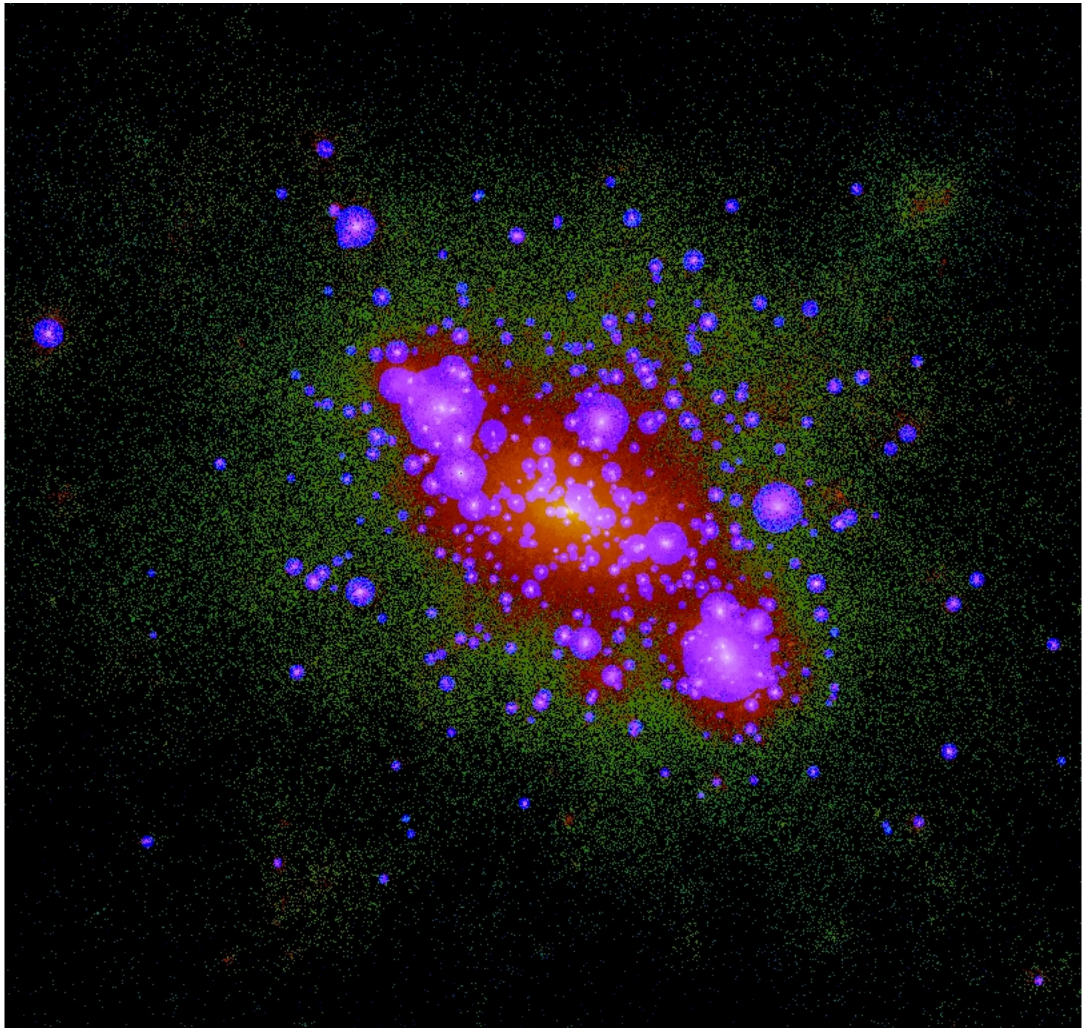
- please note that subhalo particles are included in the host halo, too!*



- subhalos are contributing to the integral properties of their hosts

*in previous versions this could be switched off, but not in the latest version anymore!





"host" halo not shown for clarity

- bottomline

AHF naturally find haloes, sub-haloes, sub-subhaloes, ...

- bottomline

AHF naturally find haloes, sub-haloes, sub-subhaloes, ...

...and has only one free parameter “ $\Delta\delta(a)$ “

(the MPI version also requires a meaningful value for LOADBALANCE_DOMAIN_LEVEL)

How to Compile?

(`DEFINEFLAGS`)

How to Compile?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/    Makefile.config
          analysis/
          bin/
          convert/
          documentation/
          Sample/
          src/
          tools/
```

HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/
    Makefile.config
    analysis/
    bin/
    convert/
    documentation/
    Sample/
    src/
    tools/
```

only ever edit this `Makefile.config`, none of the other Makefiles!

HOW TO COMPILE?

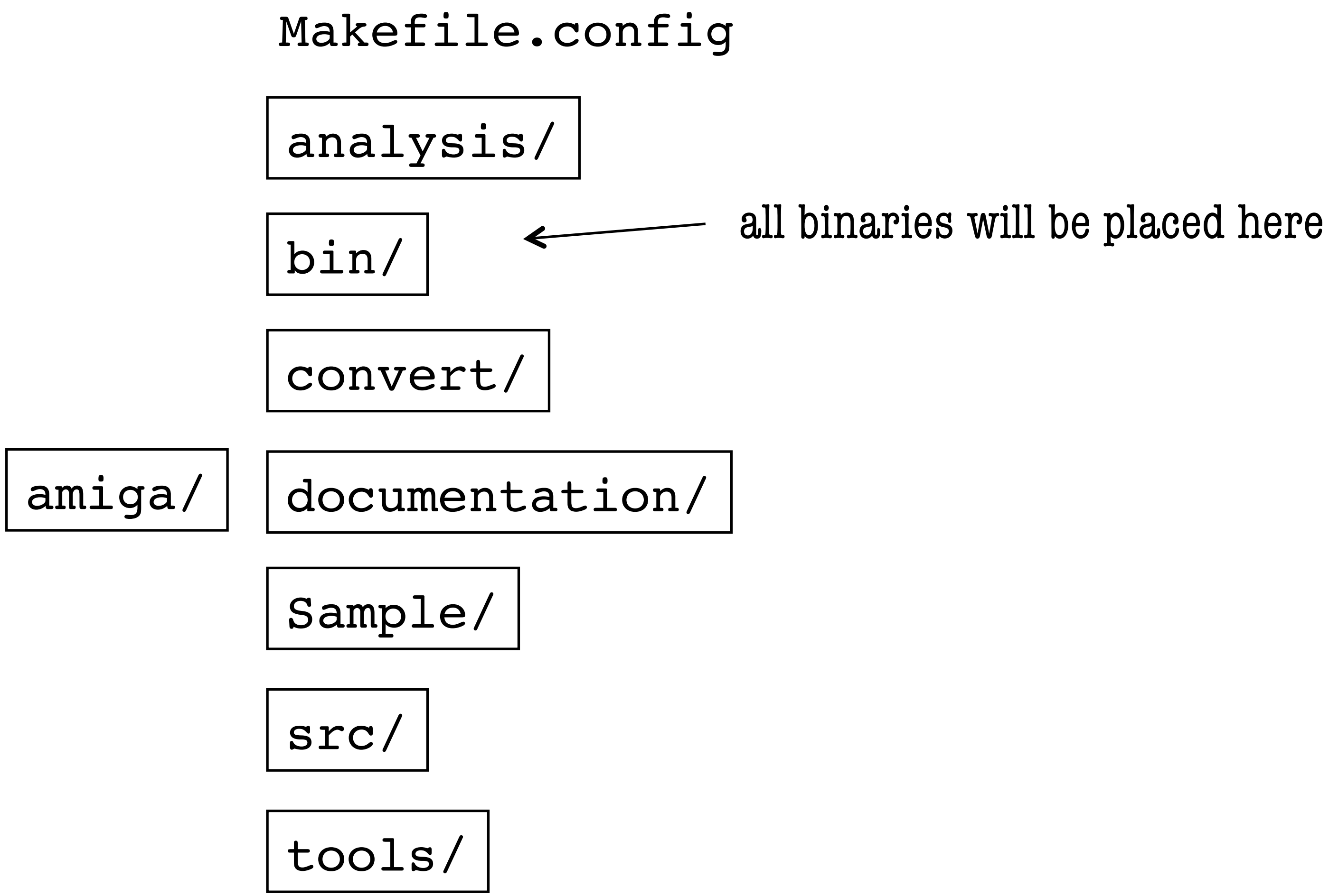
- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/    Makefile.config
          analysis/
          bin/
          convert/
          documentation/
          Sample/
          src/
          tools/
```

(analysis/) contains what is consider analysis tools, e.g. `MergerTree.c`, `HaloTracker.c`, etc...

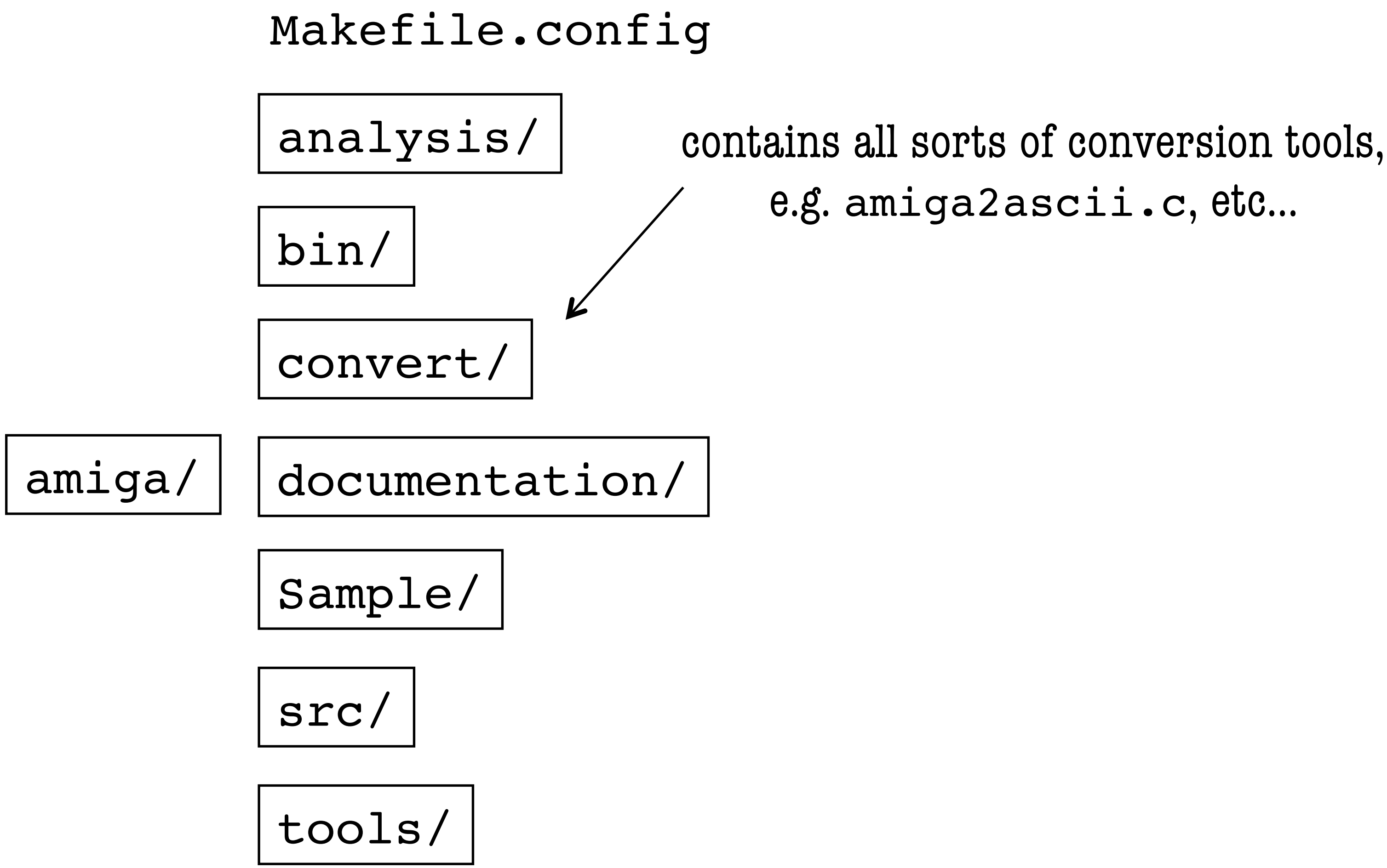
HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:



HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:



HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/   Makefile.config
         analysis/
         bin/
         convert/
         documentation/   <- all available documentation
         Sample/
         src/
         tools/
```

HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/
    Makefile.config
    analysis/
    bin/
    convert/
    documentation/
    Sample/
    src/
    tools/
```

the sample simulations need to be downloaded separately from the web page... (→ `Sample/`)

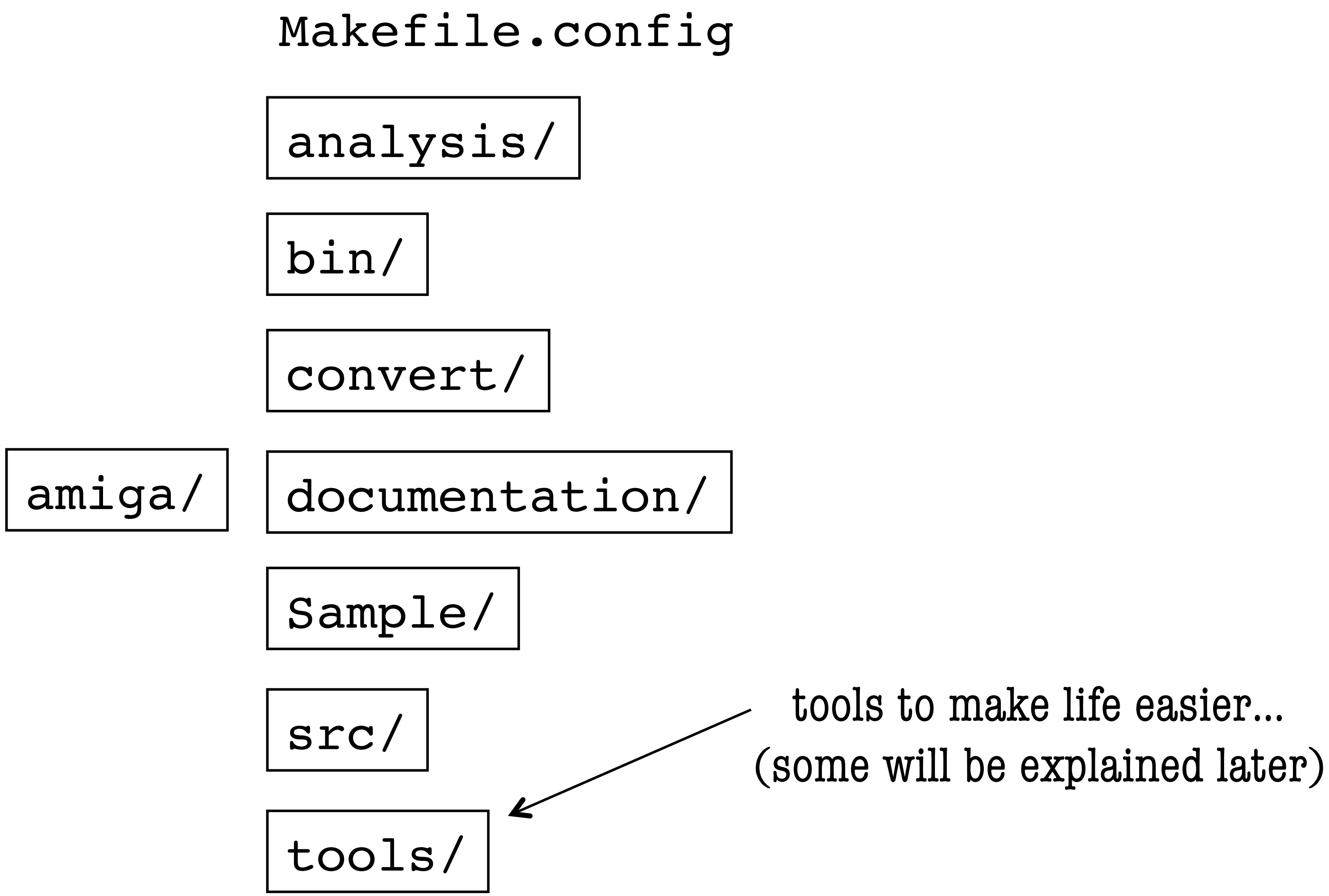
HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
amiga/
    Makefile.config
    analysis/
    bin/
    convert/
    documentation/
    Sample/
    src/        <- the heart-and-soul of AMIGA and AHF
    tools/
```

HOW TO COMPILE?

- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:



- after unpacking the tarball `amiga-v0.0.tgz` you end up with the following directory layout:

```
Makefile.config
amiga/
    analysis/
    bin/
    convert/
    documentation/
    Sample/
    src/
    tools/
```

if you plan to use ***AHF*** as a black-box the only files you ever need to touch are...

`Makefile.config`
`src/param.h`

and maybe...

`src/define.h`

`Makefile.config`
`(src/define.h)`

With the `Makefile.config` (and/or `define.h`) you decide what features to switch on or off. All features are controlled via *#ifdef FEATURE* in the source code and hence can be activated by either

-DFEATURE in the `Makefile.config` or

#define FEATURE in `define.h`

`src/param.h`

Some parameters controlling the behaviour of ***AHF*** are to be set here...

`Makefile.config`
<u>`(src/define.h)`</u>

With the `Makefile.config` (and/or `define.h`) you decide what features to switch on or off. All features are controlled via *#ifdef FEATURE* in the source code and hence can be activated by either

all available FEATURES will be explained below in the Section

DEFINEFLAGS

<u>`src/param.h`</u>

Some parameters controlling the behaviour of ***AHF*** are to be set here...

How to Compile?

Makefile.config

- Makefile.config

Please note that you need to generate a Makefile.config and should **not** touch the actual Makefile found in the top level (or any other level of the source hierarchy!) directory at all!

All your favourite flags and definitions will go into that Makefile.config and we provided a sample to be used at your leisure...

▪ OpenMP or MPI code?

Besides of the option to switch on/off various features via −DFEATURE you should also choose your system. There are three standard configurations that should work on most common machines:

- "Standard OpenMP"

  choose this for the OpenMP version

- "Standard MPI"

  choose this for the MPI version

- "Standard MPI+OpenMP"

  choose this for the MPI+OpenMP hybrid version

OpenMP and MPI work nicely together and are not mutually exclusive!

A simple `make AHFstep` will then produce the respective code...

HOW TO COMPILE?

`src/param.h`

- parameters controlling...

  - ***AHF*** behaviour
  - **GADGET** units ♭
  - MPI parallelisation

♭ **TIPSY** units, for instance, are handed to ***AHF*** differently (more later!)

- ***AHF*** behaviour MIN_NNODES

  - sets the minimum number of cells per refinement grid,
    e.g. grids containing fewer cells are not considered trustworthy...

  - controls refinements already on the `refine_grid.c` level

- ***AHF*** behaviour AHF_MINPART

  - only halos in excess of AHF_MINPART particles are written to file

    (**Note**: AHF internally stores and deals with all halos containing down to 2 particles…)

- ***AHF*** behaviour AHF_VTUNE

  - during the unbinding particles with speeds in excess of

$$v > AHF\_VTUNE \ v_{esc}$$

are considered unbound

- ***AHF*** behaviour AHF_MAX_GATHER_RAD

  - collecting particles about potential halo centres extends out to the "half-distance" of the closest refinement on the same level; this though limits this distance (in physical units!)

  - there is further an internal(!) switch that limits the distance to 1/4 of the boxsize in case you are analysing very small cosmological volumes...

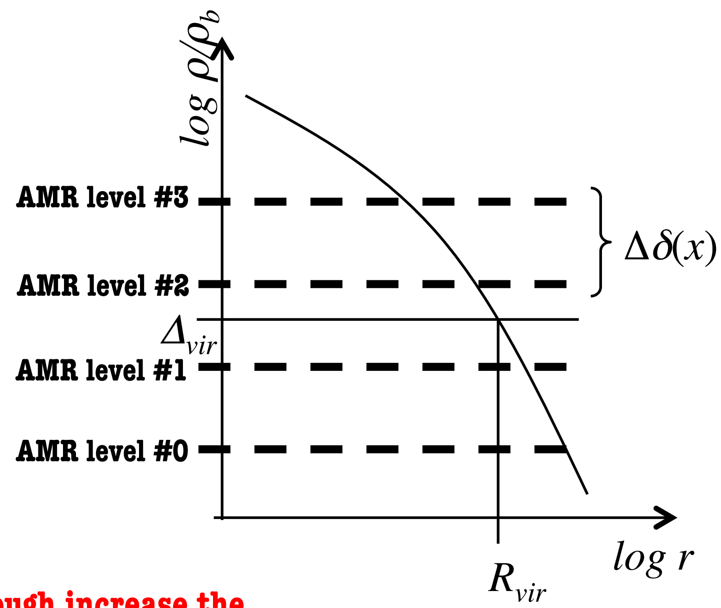
- ***AHF*** **behaviour** AHF_MIN_REF_OFFSET

  - ***AHF*** automatically determines the finest grid defining the isodensity contour closest to the virial overdensity criterion -> in the depicted example that would be AMR level #1
  - remember:

    $\Delta\delta(x)$: spacing of AMR isodensity contours as determined by refinement criterion
    Δ_{vir}: virial overdensity threshold as given by cosmology and redshift

↓

...for the depicted example
AHF would only consider AMR levels
#(1+AHF_MIN_REF_OFFSET)
(and above) in the construction of halos!



While this flag may lead to host haloes that are too small it may though increase the performance dramatically when you are only interested in subhaloes! Decide for yourself... ;-)

- ***AHF*** behaviour AHF_MASSMIX

  - when analysing multi-mass (or zoom aka re-)simulations you are dealing with all these "tidal field" particles

  - these particles can contaminate your objects and hence this parameter limits credible halos to only contain a certain fraction of mass in "tidal particles"

  - a halo is considered contaminated if the mass in high-resolution particles is less than AHF_MASSMIX and it is removed from the list.

- ***AHF*** behaviour

AHF_MAXHALO

  - when using the `AHFmaxhalo` feature (see **DEFINEFLAGS**) this sets the maximum mass a halo can have before ***AHF*** terminates

- **GADGET** support
  - GADGET_MUNIT: the mass of a **GADGET** particle
  - GADGET_LUNIT: the length unit used with the **GADGET** run

- TIPSY support

Please ignore the TIPSY parameters in *src/param.h*!

The **TIPSY** parameters in ***src/param.h*** were in use by an older version!

The handling of the **TIPSY** units for it will be explained later...

HOW TO COMPILE? src/param.h

- MPI parallelisation LOADBALANCE_DOMAIN_LEVEL

  - first of all: **THIS IS A VERY IMPORTANT PARAMETER!**

  - it sets the grid that is used to do the domain decomposition:

$$L = 2^{\mathrm{LOADBALANCE\_DOMAIN\_LEVEL}}$$

AHF farms out the particles to the desired number of CPU's and then runs a serial version of the halo finder on each of these CPU's!

Therefore, it is important to create a boundary zone on each CPU that contains (replicates of the) particles from the neighbouring cells. In order **not** to cut a halo into pieces this boundary should at least be of order the virial radius of the most massive object expected to be found within the simulation.

LOADBALANCE_DOMAIN_LEVEL hence needs to be carefully chosen, i.e. $B/2^{\mathrm{LOADBALANCE\_DOMAIN\_LEVEL}}$ should be of order that virial radius!
(where B=box size of your simulation...)

DEFINEFLAGS

- general remarks

The DEFINEFLAGS (i.e. *#ifdef FEATURE* in the code) can either be activated by using

-DFEATURE in the Makefile

or putting the desired

#define FEATURE into define.h

Makefile.config:

You will note that the Makefile.config already comes with a set of DEFINEFLAGS predefined for various projects/snapshots; and I recommend to keep track of your features in a similar way (it makes life easier when coming back to re-analyse the simulation after a vacation or any other break...)

define.h:

Please check define.h **very** carefully as some features are mutually exclusive and are being switched on or off depending on some other features! (e.g. -DGADGET automatically entails -DMULTIMASS and -DGAS_PARTICLES ...)

- classes of DEFINEFLAGS

  - ***AHF*** features
  - **GADGET** support specific flags
  - IO features
  - MPI parallelisation

rember that either -DFEATURE in `Makefile.config` or #define FEATURE in `define.h` will switch it on; however, we refer to the feature from now on as "#define FEATURE"...

- ***AHF*** features

#define AHFstep

  - ***AMIGA*** works as a stand-alone halo finder ***AHF***
  - automatically switched on when typing `make AHFstep`

- ***AHF*** features

#define AHFmaxdenscentre

- per default ***AHF*** determines the prospective halo centre as the density-weighted centre of the "end-leave" in the AMR grid tree
- this feature rather uses that cell in the end-leave grid with the highest density value as prospective halo centre

- ***AHF*** features

#define AHFpotcentre

  - per default ***AHF*** determines the prospective halo centre as the density-weighted centre of the "end-leave" in the AMR grid tree
  - this feature rather uses that cell with the lowest value of the potential as the potential halo centre
  - **Note:** this feature requires substantially more time for ***AHF*** to run as it solves for the potential on the complete AMR hierarchy!

- ***AHF*** features

#define AHFgeomcentre

  - per default ***AHF*** determines the prospective halo centre as the density-weighted centre of the “end-leave” in the AMR grid tree
  - this feature rather uses the geometrical centre of the refinement patch

- ***AHF*** features

#define AHFcomcentre

- per default ***AHF*** determines the prospective halo centre as the density-weighted centre of the "end-leave" in the AMR grid tree

- this feature rather uses the centre-of-mass of the particles encompassed by the refinement patch

some trial-and-error with these AHF***centre **flags indicated that** AHFcomcentre **gives the best results for subhaloes...at least for our simulations...**

- ***AHF*** features

#define AHFmaxhalo

  - once a halo contains in excess of AHF_MAXPART particles ***AMIGA*** will terminate

  - only useful when running a simulation with ***AMIGA*** and performing on-the-fly halo analysis

- ***AHF*** features

#define AHFnoHubbelDrag

  - will not consider the Hubble drag +***H*r*** during unbinding

- ***AHF*** features

#define AHFptfocus=*value*

  - only keeps particles of a certain kind for AHF analysis

  - set the "particle-type-to-keep" as follows:

    0 = gas particles

    1 = dark matter particles

    4 = star particles

  - if you have more than one dark matter type, please consult `main.c` where this feature is to be found and/or get in touch with us...

- ***AHF*** features

#define WITH_AHF_HALOS_SFC

- a complete rehash of the way gather_hostParts() works:

  You may have experienced that ***AHF*** slows down at higher redshifts and/or gets stuck in the routine gather_hostParts()?!

  If not, you are lucky; if yes, this flag may be the solution...

  Here we utilize the Peano-Hilbert curve to collect those particles within the initial gathering radius of each object which avoids the otherwise badly coded loops over (nearly) all particles...

- ***AHF*** features

  #define WITH_AHF_HALOS_SFC

  - a complete rehash of the way gather_hostParts() works:

    You may have experienced that ***AHF*** slows down at higher redshifts and/or gets stuck in the routine gather_hostPart…

    If not, you…

    This feature *dramatically* **increases the performance of AHF and hence is currently** *hardwired* **!**

    …Hilbert curve to collect those particles …the initial gathering radius of each object which avoids the otherwise badly coded loops over (nearly) all particles...

HOW TO COMPILE?

- ***AHF*** features

#define AHFnoremunbound

  - skips the unbinding procedure

- ***AHF*** features

#define MANUAL_DVIR=*value*

  - lets you set the virial overdensity value manually
  - the *value* will be used as Δ_{vir} in the virial radius determination
  - check calc_virial() in `cosmology.c` to adjust it to your needs

- ***AHF*** features

#define AHFreducedinertiatensor

  - uses the reduced moment of inertia tensor to determine halo shapes

- ***AHF*** features

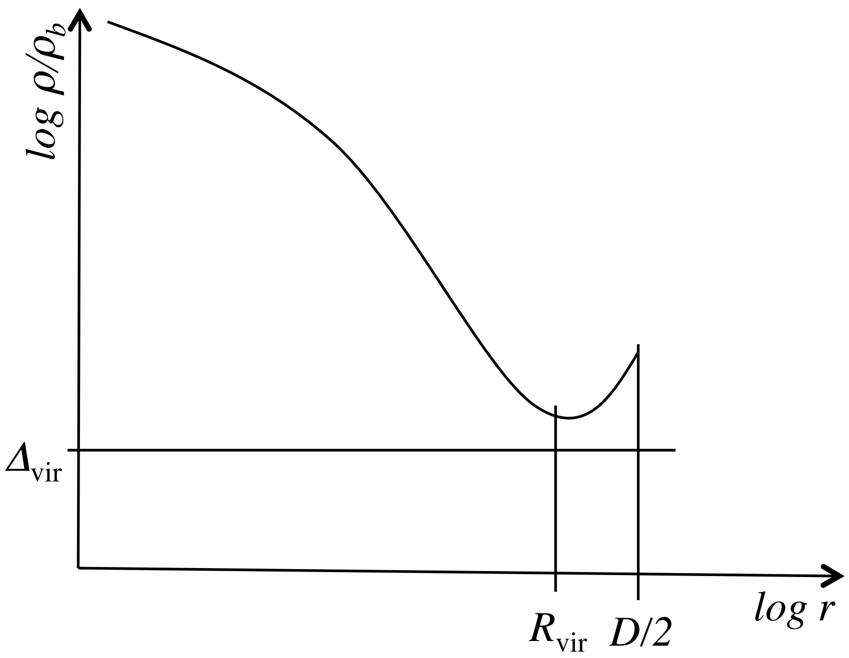
#define AHFabsangmom

  - dump absolute angular momentum rather than $\vec{L}/|\vec{L}|$

- ***AHF*** features

#define AHFsplinefit

  - uses a splinefit routine to determine R_{vir}



- ***AHF*** features

#define AHFcentrefile

  - writes an additional file containing all the prospective halo centres, i.e. the density peaks found in the simulation

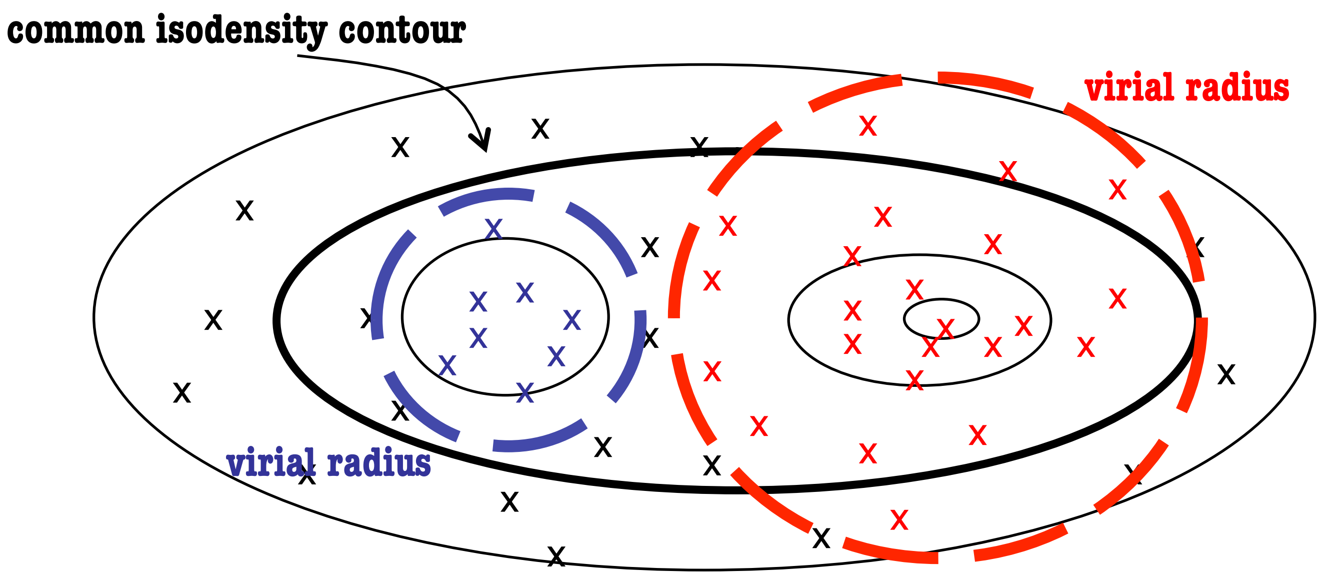
- ***AHF*** features

#define AHFsubstructure

  - writes an additional file containing information about which halo is a subhalo of what host
  - the standard implementation looks for subhalos via an N^2-loop checking whether a halo lies within the virial radius of another halo

- ***AHF* features** #define AHFgridsubstructure

  - writes an additional file containing information about which halo is a subhalo of what host
  - works only together with #define AHFsubstructure
  - this implementation defines substructure as those objects that lie within common isodensity contours



AHFsubstructure would not consider **halo** as subhalo of **halo** while AHFgridsubstructure will!

- ***AHF*** features

#define AHFphspdens

  - writes elaborate information about the phase-space density into the *.AHF_profiles file (in addition to the standard info...)

- ***AHF*** features

#define GAS_PARTICLES

  - in case you are supplying also gas and star particles ***AHF*** will add additional columns to the *.AHF_halos file containing information about the properties of the gas and stellar content of each halo alone...

Note: you cannot switch off this feature for star particles, i.e. GAS_PARTICLES switches it on for both!

- ***AHF* features** #define GAS_PARTICLES

  - in case you are supplying also gas and star particles ***AHF*** will add additional columns to the *.AHF_halos file containing information about the properties of the gas and stellar content of each halo alone...

This feature should *definitely* be used whenever you are dealing with simulations including baryons (gas and/or stars)!

Note: you cannot switch off this feature for star particles, i.e. GAS_PARTICLES switches it on for both!

- ***AHF*** features

#define AHFverbose

  - increase the verbosity of ***AHF*** dramatically:

    you will now find information for each halo as it is being processed in the logfile of ***AHF***

- ***AHF*** features

  - there are three features that control halo vs. subhalo treatment:

```
#define PARDAU_DISTANCE
#define PARDAU_NODES
#define PARDAU_PARTS
```

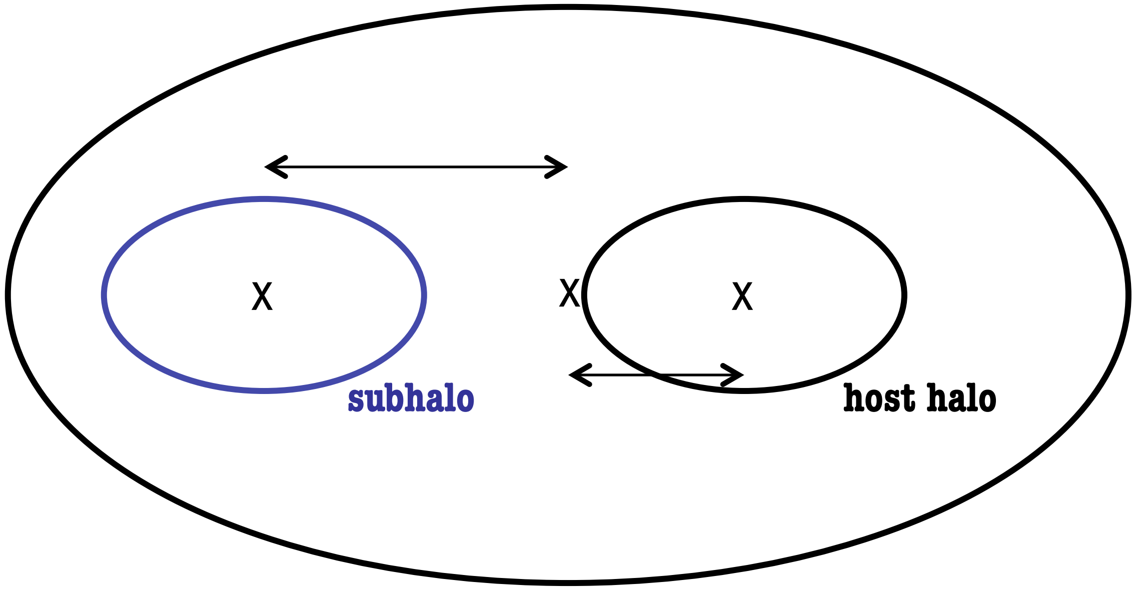
  - they control the classification into halo, subhalo, sub-subhalo, etc.

  - a major merger of two nearly equal mass objects can cause a lot of trouble and hence experimenting with this feature in that case may help?!

#define PARDAU_DISTANCE

- ***AHF* features**

  - parent-daughter assignment is done by distance

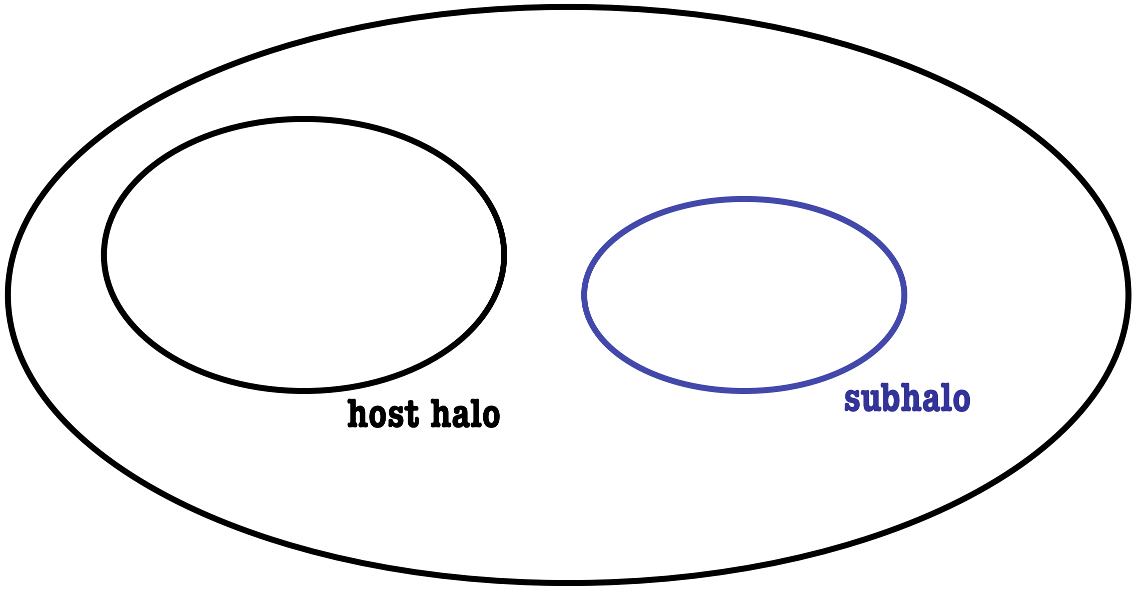


  - those parent-daughter grids with the smallest distance are being tagged as "trunk" in the AMR grid tree

▪ *AHF* features

#define PARDAU_NODES

- parent-daughter assignment is done by number of cells

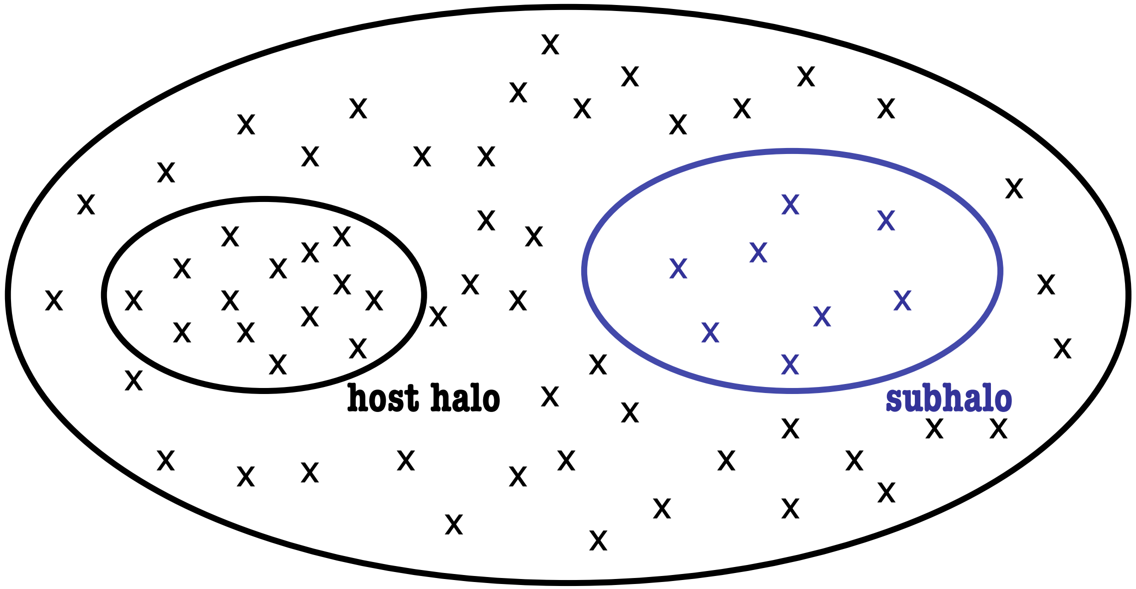


- the largest daughter grid is being tagged as “trunk” in the AMR grid tree

- ***AHF*** **features**

#define PARDAU_PARTS

- parent-daughter assignment is done by number of particles



- the daughter grid with the most particles is being tagged as "trunk" in the AMR grid tree

- this daughter grid is the most likely candidate for further refinement and encompassing the highest density peak, respectively

switched on by default (cf. `define.h`)

- ***AHF*** **features** #define PARTICLES_INFO

  - dumps information about particle type (DM, gas, star) into `*.AHF_particles` file as additional columns next to the id:

0	gas particle
1	dark matter particle
2	(not used)
3	heavy dark matter particle
4	star particle

  - this feature is obviously tailored for the analysis of some special runs and hence may be of only limited use for the "black-box" user...

- **GADGET** support #define GADGET_IDS

  - stick to the particle id's as found in the **GADGET** file and drag them through to the `*.AHF_particles` output file

  - probably the best option when analysing **GAGDET** simulations as it will be **your** responsibility to make sense out of the id's in the end ;-)

- **GADGET** support

#define GADGET_LUNIT_KPC

  - assumes that the length unit in the **GADGET** file is kpc/h

- **TIPSY** support #define TIPSY_ZOOMDATA
  - shifts TIPSY particles by half-a-boxsize when reading

- IO features

#define BYTESWAP

  - forces a byteswap when reading the simulation binary file
  - you need to use this flag when...
    - » your data is little_endian but your analysis machine big_endian
    - » your data is big_endian but your analysis machine little_endian

Note: this feature is obsolete when analysing **GADGET** files with the MPI version

- MPI parallelisation #define WITH_MPI
  - now ***AHF*** can be run on a distributed memory machine
  - please refer to the additionally supplied `MPI.txt` for more details!

- MPI parallelisation

#define NEWSTARTRUN

  - the MPI version uses a completely new way for
    - reading in data (cf. `libio/` in `src/`)
    - starting the simulation
  - WITH_MPI is only function with NEWSTARTRUN
  - NEWSTARTRUN is functional without WITH_MPI though...

- MPI parallelisation

#define NEWSTARTRUN

  - the MPI version uses a completely new way fo...
    - reading in data (...
  - W...
  - NE... ...without WITH_MPI though...

This feature should *definitely* be used and hence we hardwired it!
If you prefer to remove it, we suggest you contact us beforehand...

- OpenMP parallelisation #define WITH_OPENMP
  - the processing of individual halos will be cast to different threads
  - define # of threads via OMP_NUM_THREADS environment variable
  - works perfectly together with WITH_MPI

- miscellaneous

#define NGRID_MAX

  - sets the maximal allowed refinement level L_{max}

How to Run?

1. make the binary `AHFstep`

(see "how to compile" section...)

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

full name (including path) of the snapshot you wish to analyse

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed *iFormat* *NprocRead*
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

iFormat:

0	***AMIGA*** binary
10	ASCII binary
20	**CubeP3M** binary
60	**GADGET** binary (single snapshot)
61	**GADGET** binary (multiple snapshots)
80	single precision DEVA binary
81	native DEVA binary
90	TIPSY binary

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

number of processors reading:

- the number of processors used to analyse the data and the number used to read in the data can be different!
- even if you are not using the MPI version, please provide a dummy number here!

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

`AHFstep` will write several output files all lead by this prefix, e.g.

```
prefix.AHF_halos
prefix.AHF_profiles
prefix.AHF_particles
...
```

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed *iFormat* *NprocRead*
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

number of cells (in 1D) for the regular grid (i.e. domain grid) covering the whole computational domain

(rule of thumb: $L^3 \approx (2N)^3$ or N^3)

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain ← refinement criterion (= number of particles per cell) on the domain grid
RefCritAMR
0
0
0
0

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR ← refinement criterion (= number of particles per cell) on all refinement grids
0
0
0
0

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

(RefCritDomain, RefCritAMR): something between 3 – 6 should be fine...

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

unimportant numbers for `AHFstep`!
(but they nevertheless need to be set...)

2. prepare `AHFstep.input` with the following information:

NameOfSnapshotToBeAnalysed iFormat NprocRead
PrefixForOutputFiles

Note:

When compiling the MPI version of `AHFstep` be aware that there is one parameter in `param.h` (i.e. LOADBALANCE_DOMAIN_LEVEL) that needs to be set wisely!

0
0

3. executing `AHFstep`

- OpenMP/serial version
  - set OMP_NUM_THREADS (OpenMP version only)
  - type `AHFstep AHFstep.input`

- MPI version
  - type `mpiexec -n NprocRun AHFstep AHFstep.input`

Note, the number of processors NprocRun used to analyse the data and the number NprocRead used to read in the data can be different!

3. executing `AHFstep`

- OpenMP/serial version
  - set OMP_NUM_THREADS (OpenMP version only)
  - type `AHFstep AHFstep.input`

Note that the OpenMP and MPI version function well together and are not mutually exclusive!

- MPI version
  - type `mpiexec -n NprocRun AHFstep AHFstep.input`

Note, the number of processors NprocRun used to analyse the data and the number NprocRead used to read in the data can be different!

Supported Input File Formats

SUPPORTED INPUT FILE FORMATS

- remember `AHFstep.input`:

NameOfSnapshotToBeAnalysed *iFormat* *NprocRead*
PrefixForOutputFiles
L
RefCritDomain
RefCritAMR
0
0
0
0

iFormat:

0	***AMIGA*** binary
10	ASCII binary
20	**CubeP3M** binary
60	**GADGET** binary (single snapshot)
61	**GADGET** binary (multiple snapshots)
80	DEVA binary
90	TIPSY binary

- **GADGET** units
  - edit `src/param.h`:
    - GADGET_MUNIT: the mass of a **GADGET** particle
    - GADGET_LUNIT: the length unit used with the **GADGET** run

▪ GADGET DEFINEFLAGS

#define GADGET_IDS

- stick to the particle id's as found in the **GADGET** file and drag them through to the `*.AHF_particles` output file

- probably the best option when analysing **GAGDET** simulations as it will be **your** responsibility to make sense out of the id's in the end ;-)

#define GADGET_LUNIT_KPC

- assumes that the length unit in the **GADGET** file is kpc/h

▪ TIPSY units

Unfortunately a TIPSY binary file does not store any information about the cosmology and/or units and hence the user has to specify them somewhere...and we decided that this should go into a file *tipsy.info* !

example for *tipsy.info*

0.26	Ω_0	
0.74	Λ_0	
20.0	the box size	(in Mpc/h)
690.988298942671	the velocity unit	(in km/sec)
2.2197e15	the mass unit	(in $M_\odot$/h)

- This file should be found in the same directory from which ***AHF*** is run.
- Please note that the boxsize and mass are in 1/h units which is important!
- If we understand correctly, the velocity unit is H_0*B_0/sqrt(8piG), however, we rather ask the user to provide this information than calculating it ourselves...

Format of Output Files

FORMAT OF OUTPUT FILES *.AHF_halos

- integral properties

(1) npart	number of particles in halo	
(2) nvpart	mass of halo in internal units	
(3) Xc		
(4) Yc	position of halo	[Mpc/h]
(5) Zc		
(6) VXc		
(7) Vyc	peculiar velocity of halo	[km/sec]
(8) VZc		
(9) Mvir		[$M_\odot$/h]
(10) Rvir	virial radius	[kpc/h]
(11) Vmax	maximum of rotation curve	[km/sec]
(12) Rmax	position of rotation curve maximum	[kpc/h]
(13) sigV	3D velocity dispersion	[km/sec]
(14) lambda	spin parameter (Bullock et al. 2001 definition)	

FORMAT OF OUTPUT FILES *.AHF_halos

- integral properties

(15) Lx		
(16) Ly	orientation of angular momentum vector	$\|L\|=1$
(17) Lz		
(18) a	largest axis (derived from inertia tensor, normalized to unity)	
(19) Eax		
(20) Eay	orientation of corresponding axis	$\|Ea\|=1$
(21) Eaz		
(22) b	second largest axis (b/a)	
(23) Ebx		
(24) Eby	orientation of corresponding axis	$\|Eb\|=1$
(25) Ebz		
(26) c	third largest axis (c/a)	
(27) Ecx		
(28) Ecy	orientation of corresponding axis	$\|Ec\|=1$
(29) Ecz		

FORMAT OF OUTPUT FILES *.AHF_halos

- integral properties

(30) ovdens	overdensity at virial radius	
(31) Redge	actual edge of the halo (ignore! not fully implemented)	
(32) nbins	number of bins used for the *.AHF_profiles file	
(33) Ekin	kinetic energy	$[M_\odot/h\ (km/sec)^2]$
(34) Epot	potential energy	$[M_\odot/h\ (km/sec)^2]$
(35) mbp_offset	offset between most bound particle and halo centre	[kpc/h]
(36) com_offset	offset between centre-of-mass and halo centre	[kpc/h]
(37) r2	position where $\rho\, r^2$ peaks	[kpc/h]
(38) lambdaE	classical spin parameter (Peebles' definition)	
(39) v_esc	escape velocity at Rvir	[km/sec]

All these values have been derived using **all** particles inside the halo, i.e. dark matter, gas, and star particles (if present)...

Further, all properties are in **comoving** coordinates!

FORMAT OF OUTPUT FILES *.AHF_halos

- integral properties

(30) ovdens	overdensity at virial radius	
(31) Redge	actual edge of the halo (warning: not fully implemented)	
(32) nbins	number of bins used for the *.AHF_profiles file	
(33) Ekin	kinetic energy	$[M_\odot/h\ (km/sec)^2]$
(34) Epot	potential energy	$[M_\odot/h\ (km/sec)^2]$
(35) mbp_offset	offset between most bound particle and halo centre	[kpc/h]
(36) com_offset	offset between centre-of-mass and halo centre	[kpc/h]
(37) r2	position where ρr^2 peaks	[kpc/h]
(38) lambdaE	classical spin parameter (Peebles' definition)	
(39) v_esc	escape velocity at Rvir	[km/sec]

In case you compiled with the feature GAS_PARTICLES you will find even more columns where some of the properties have been determined for the gas and star particles *alone* !

We hope that the columns will be self-explanatory!? If in doubt, please contact us...

All these values have been derived using **all** particles inside the halo, i.e. dark matter, gas, and star particles (if present)...

Further, all properties are in **comoving** coordinates!

- radial profile of selected properties

(1) r	right edge of radial bin	[kpc/h]
(2) npart	number of particles inside sphere of radius r	
(3) nvpart	mass inside sphere of radius r	[internal units]
(4) ovdens	$M(<r)/(4\pi r^3/3)$ / ρ_b	
(5) dens	$M(r)/(4\pi r^3/3)$ / ρ_b with $M(r)$ = mass in current *shell*	
(6) vcirc	rotation curve	[km/sec]
(7) sigv	velocity dispersion of material inside r-sphere	[km/sec]
(8) Lx		
(9) Ly	angular momentum of material inside r-sphere	
(10) Lz		[$M_\odot$/h Mpc/h km/sec]

Note:

a negative value for "(1) r" indicates that the results at that radius have not converged and are dominated by two-body collisions according to the criterion of Power et al. (2003)

- radial profile of selected properties

(11) a	largest axis (derived from inertia tensor, normalized to unity)	
(12) Eax		
(13) Eay	orientation of corresponding axis	|E|=1
(14) Eaz		
(15) b	second largest axis (b/a)	
(16) Ebx		
(17) Eby	orientation of corresponding axis	|E|=1
(18) Ebz		
(19) c	third largest axis (c/a)	
(20) Ecx		
(21) Ecy	orientation of corresponding axis	|E|=1
(22) Ecz		
(23) Ekin	kinetic energy of material inside r-sphere	[$M_\odot$/h (km/sec)^2]
(24) Epot	potential energy of material inside r-sphere	[$M_\odot$/h (km/sec)^2]
(25) v_esc	escape velocity from material inside r-sphere	[km/sec]

- radial profile of selected properties

(11) a	largest axis (derived from inertia tensor, normalized to unity)	
(12) Eax		
(13) Eay	orientation of corresponding axis	|E|=1
(14) Eaz		
(15) b	second largest axis (b/a)	
(16) Ebx		
(17) Eby	orientation of corresponding axis	|E|=1
(18) Ebz		
(19) c	third largest axis (c/a)	
(20) Ecx		
(21) Ecy	orientation of corresponding axis	|E|=1
(22) Ecz		
(23) Ekin	kinetic energy of material inside r-sphere	[$M_\odot$/h (km/sec)^2]
(24) Epot	potential energy of material inside r-sphere	[$M_\odot$/h (km/sec)^2]
(25) v_esc	escape velocity from material inside r-sphere	[km/sec]

In case you compiled with the feature GAS_PARTICLES you will find even more columns where some of the properties have been determined for the gas and star particles *alone* !

Again, if in doubt, get in touch with us...

FORMAT OF OUTPUT FILES *.AHF_particles

- access to the particles in a halo

N1	N1 = number of particles in halo #1
id1	
id2	N1 id's of those particles belonging to halo #1
...	
idN1	
N2	N2 = number of particles in halo #1
id1	
id2	N2 id's of those particles belonging to halo #2
...	
idN2	
N3	N3 = number of particles in halo #1
id1	
id2	N3 id's of those particles belonging to halo #3
...	
idN3	
etc.	

- access to the particles in a halo
  - some notes on the id's:
    - » id's start at zero (*C* convention)
    - » only dark matter particles have unique throughout a simulation
    - » both `MergerTree.c` and `HaloTracker.c` rely on unique id's
    - » check carefully how you read the particles and their id's

- access to the particles in a halo
  - some notes on the id's:
    - » id's start at zero (*C* convention)
    - » only dark matter particles have unique id's throughout a simulation
    - » both `MergerTree.c` and `HaloTracker.c` rely on unique id's
    - » check carefully how you read the particles and their id's

There is a feature `-DPARTICLES_INFO` that leads to a marginally different `*.AHF_particles` file that contains additional information about the actual particle types:

0: gas particle
1: DM particle
4: star particle

Toolbox

Toolbox

- MergerTree
- HaloTracker

AHF - Amiga's Halo Finder

MergerTree.c

- how to compile?
  - simply type `make MergerTree`
- how to run?
  - execute `bin/MergerTree`
  - you will be prompted for a number of things:

HowManyFiles

NamesOfParticlesFiles

NameForOutputFiles

- how to compile?
  - simply type `make MergerTree`
- how to run?
  - execute `bin/MergerTree`
  - you will be prompted for a number of things:

HowManyFiles ← the cross-correlation will be done between two *.AHF_particles files and hence this number should always be > 2, obviously...

NamesOfParticlesFiles

NameForOutputFiles

- how to compile?
  - simply type `make MergerTree`
- how to run?
  - execute `bin/MergerTree`
  - you will be prompted for a number of things:

HowManyFiles

NamesOfParticlesFiles ← here you need to provide the names of thos files for which you like to have the cross-correlation done...

NameForOutputFiles

here you need to provide the names of thos files for which you like to have the cross-correlation done...
if *HowManyFiles* > *2* the correlation will be done for
File1 –> File2
File2 –> File3
File3 –> File4
etc.

- how to compile?
  - simply type `make MergerTree`
- how to run?
  - execute `bin/MergerTree`
  - you will be prompted for a number of things:

HowManyFiles

NamesOfParticlesFiles

NameForOutputFiles

you further need to supply names for the output files. the correlation between two files will be written into one "mtree" file and hence you need to specify *HowManyFiles-1* names...

- how to compile?
  - simply type `make MergerTree`
- how to run?
  - execute `bin/MergerTree`
  - you will be prompted for a number of things:

HowManyFiles

NamesOfParticlesFiles

NameForOutputFiles

you further need to supply names for the output files. the correlation between two files will be written into one "mtree" file and hence you need to specify *HowManyFiles-1* names...

and how does `MergerTree` work?

- `MergerTree` solely relies on the particle id's as found in *.AHF_particles

- it steps through each halo present in the file #1

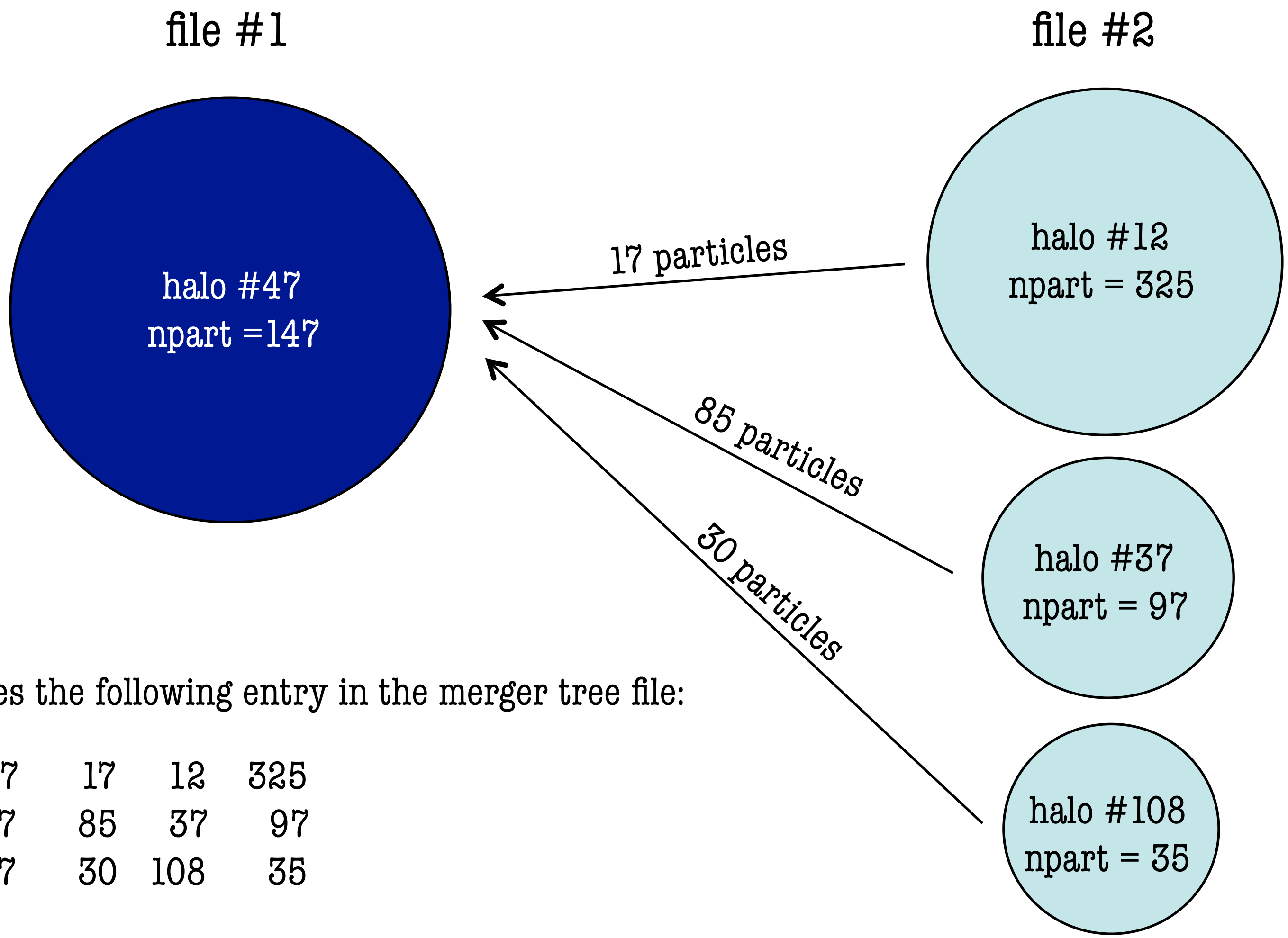
- it locates all its constituent particles in the file #2

- it keeps track of:
  - halos in file #2 sharing particles with that halo from file #1
  - the actual number of shared particles

- it writes two output files:
  - one file containing the complete merger tree information: *NameForOutputFile*
  - one file providing a quick link to the "father": *NameForOutputFile*_idx

both files will be explained in more detail now...

- merger tree for a sample halo in file #1



this gives the following entry in the merger tree file:

47	147	17	12	325
47	147	85	37	97
47	147	30	108	35

...and the following entry in the *_idx file:

47	37

- Notes and Hints

  - the sum of all "shared" particles (middle column) does not need to add up to the total number of particles in the halo as we ignore halos below a certain mass threshold (both in ***AHF*** as well as in `MergerTree`)

  - the most massive progenitor (cf. fifth column) is not necessarily the actual "father" halo; we tag that progenitor as "father" that shares the most particles with the actual halo in file #1. i.e. in the example halo #37 in file #2 will be considered the father!

  - the situation becomes quite complicated for subhalos as they share all their particles with a) their father and b) the host (if you chose to analyse using the AHF2 feature!); we though tried our best to capture these instances and deal with it...

- Notes and Hints

  - file #1 and file #2 do not necessarily need to be snapshots at different times of the same simulation; you can also do a cross-correlation between different simulations run with the same phases (e.g. CDM vs. WDM)...

  - the fastest way to get information about “who is a subhalo of who” is by running MergerTree “on itself”, i.e. create a merger tree of only one `*.AHF_particles` files with itself.

HaloTracker.c

Halo

HaloTracker is no longer supported!
If you plan to use it, get in touch...

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID ← which halo do you like to track? either give its id or type -1 for all halos...

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

`HaloTracker` obviously requires one ***AHF*** analysis as input. It will use the information about halos in there for tracking...
Note that `HaloTracker` uses both the `*.AHF_halos` and `*.AHF_particles` files; the particle id's are important for the tracking while it also requires the halo centres upon startup. You should though only provide the prefix as the tracker constructs the names itself...

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

Given an ***AHF*** analysis the `HaloTracker` follows individual particles throughout a series of snapshots, i.e. it reads the full binary snapshot file and basically performs a new ***AHF*** analysis of it...
Here you provide the number of snapshots it should plough through...

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots ← ...and here you give their names.

PrefixForOutputFiles

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

For each snapshot `HaloTracker` will write one output file in the same format as an ***AHF*** analysis file!
They though will be called
`*.TRK_halos`, `*.TRK_profiles`, etc.
Here you provide the prefix...

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things:

HaloID

PrefixOfAHFfiles

HowManySnapshots

NamesOfSnapshots

PrefixForOutputFiles

For each snapshot `HaloTracker` will write one output file in the same format as an ***AHF*** analysis file!
They though will be called `*.TRK_halos`, `*.TRK_profiles`, etc.
Here you provide the prefix...

Note: the format of the `HaloTracker` and ***AHF*** files is indistinguishable!

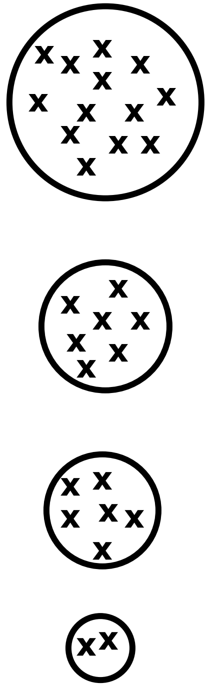
- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things...
- advantages of `HaloTracker` over ***AHF***
  - halo #N will **always** be halo #N throughout all files!
  - hence there is no need for `MergerTree` anymore!
- disadvantage of `HaloTracker`
  - only mass loss is taken into account
  - mergers have to be worked out manually (quite a pain!)

- how to compile?
  - simply type `make HaloTracker`
- how to run?
  - execute `bin/HaloTracker`
  - you will be prompted for a number of things…
- advantages of `HaloTracker` over ***AHF***
  - halo #N will **always** be halo #N throughout all files!
  - hence there is no need for `MergerTree` anymore!
- disadvantage of `HaloTrack`
  - … be worked out manually (quite a pain!)

`HaloTracker` is only well suited for the study of satellite dynamics and tidal debris fields…

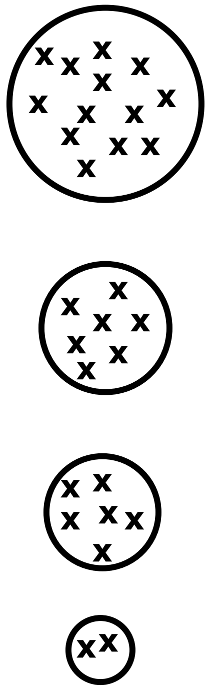
- mode of operation

simulation at time t



- mode of operation

simulation at time t



this information is...

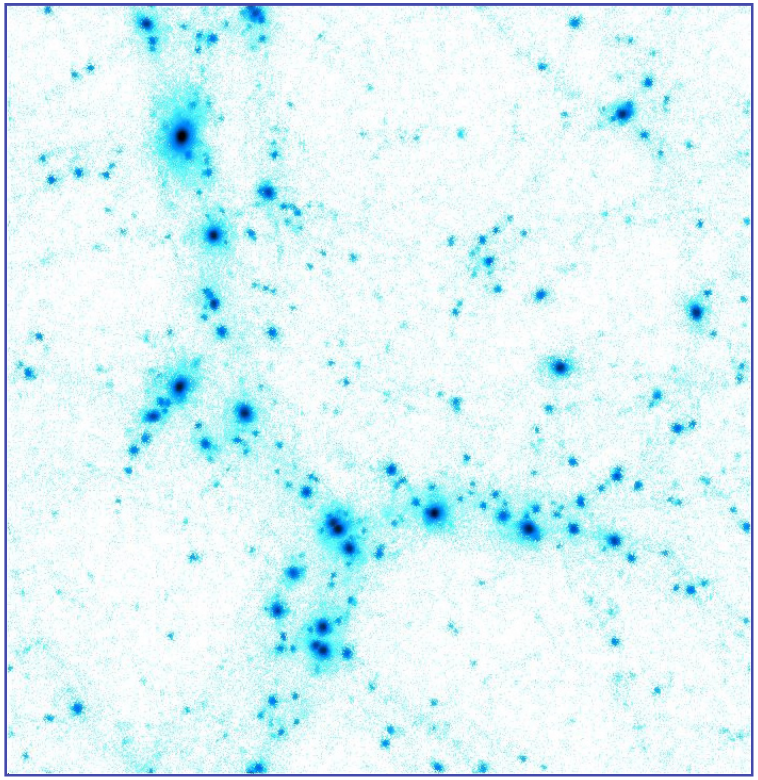
- initially provided by an ***AHF*** analysis

  and after that by the

- tracker analysis of the previous time step

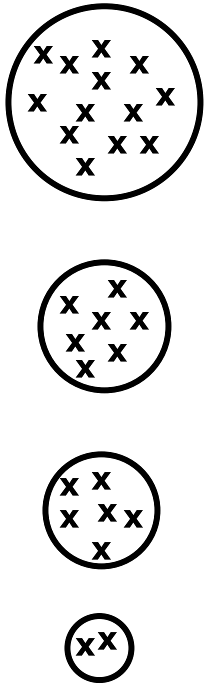
- mode of operation

simulation at time $t+\Delta t$



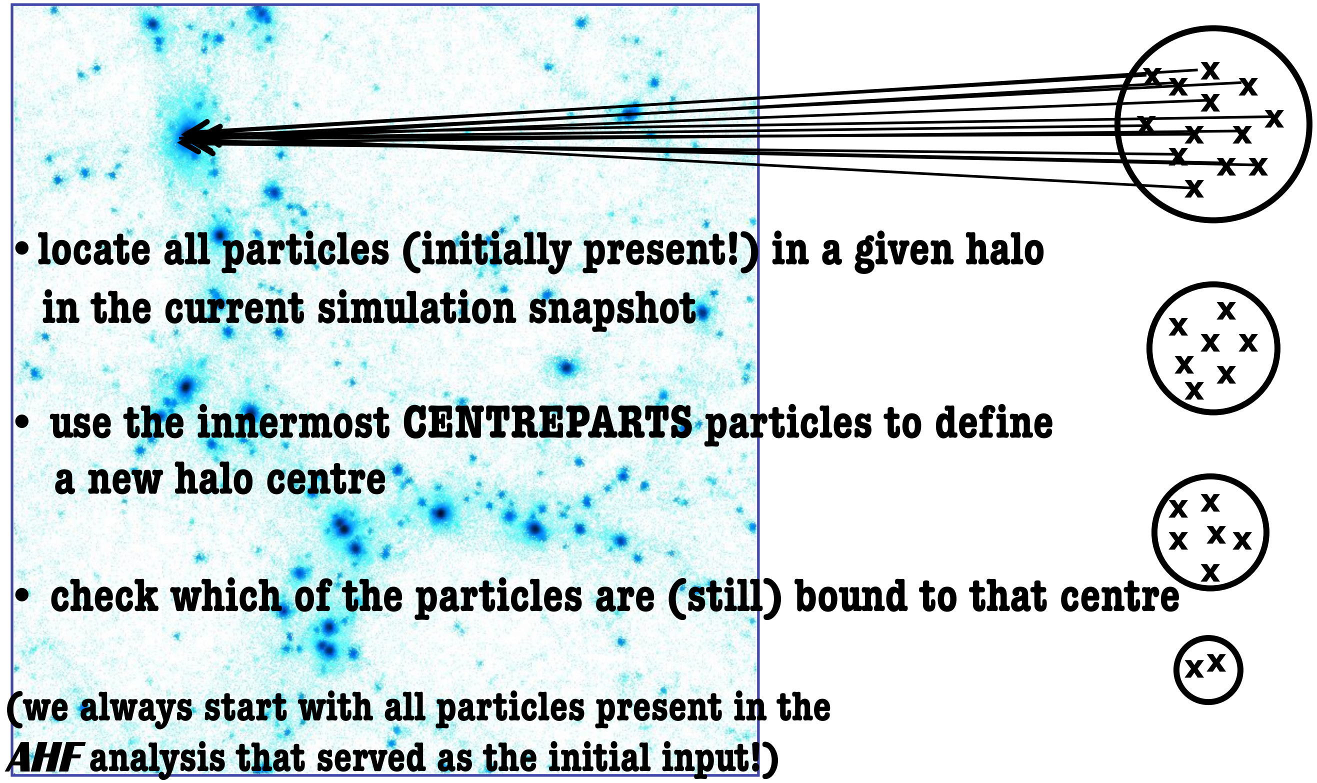
(simulation snapshot)

simulation at time t



- mode of operation

simulation at time $t+\Delta t$ simulation at time t



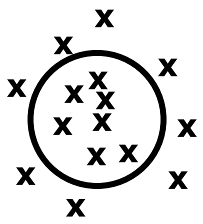
- **locate all particles (initially present!) in a given halo in the current simulation snapshot**
- **use the innermost CENTREPARTS particles to define a new halo centre**
- **check which of the particles are (still) bound to that centre**

(we always start with all particles present in the *AHF* analysis that served as the initial input!)

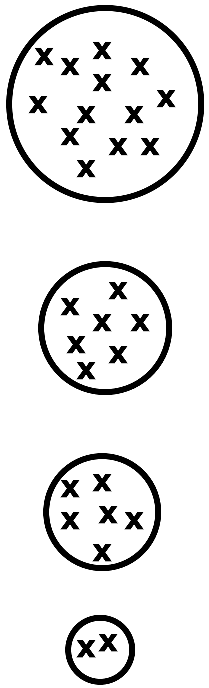
(simulation snapshot)

- mode of operation

simulation at time $t+\Delta t$



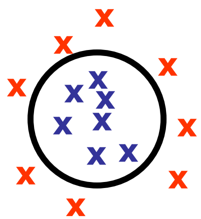
simulation at time t



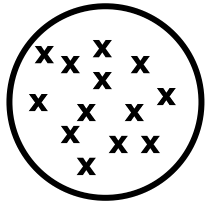
- **determine new halo properties**

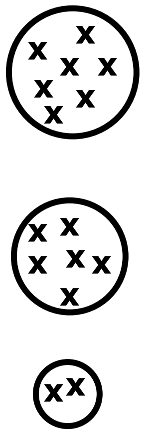
- mode of operation

simulation at time $t+\Delta t$



simulation at time t





- **determine new halo properties (based upon bound particles)**
- **keep track of debris field**

- parameters
  - CENTREPARTS — number innermost particles used for centre determination
  - TRK_MINPART — consider halo destroyed if it contains less particles
  - TRK_VTUNE — same as AHF_VTUNE, i.e. particles are unbound if

$$v > TRK\_VTUNE \;\; v_{esc}$$

- features
  - #define DEBRIS — also write a file containing debris particles
  - #define SAVE_IDS — always use initially present particles when checking for boundness to new halo centre (without this feature you only check the which of the previously bound particles remain bound; you further won't be able to track the complete debris field)