FLASH Code Tutorial

part III
sink particles & feedback

Robi Banerjee
Hamburger Sternwarte
banerjee@hs.uni-hamburg.de

FLASH code: Sinks

Motivation:

- modelling of dense regions in collapse simulations, e.g. star formation (first introduced by *Mathew Bates* 1995 in SPH simulations)
- "controlled" violation of the Truelove criterion (*Truelove et al.* 1997)

  $\rightarrow \lambda_J > 4 \times \Delta x$

- allows **long-term** runs of star forming regions with binaries, stellar clusters, discs, self-consistent outflows, ...
- source for sub-grid feedback models: (mass, accretion rate, spin, ... are known) $\Longrightarrow$ winds, radiation, SN, etc.

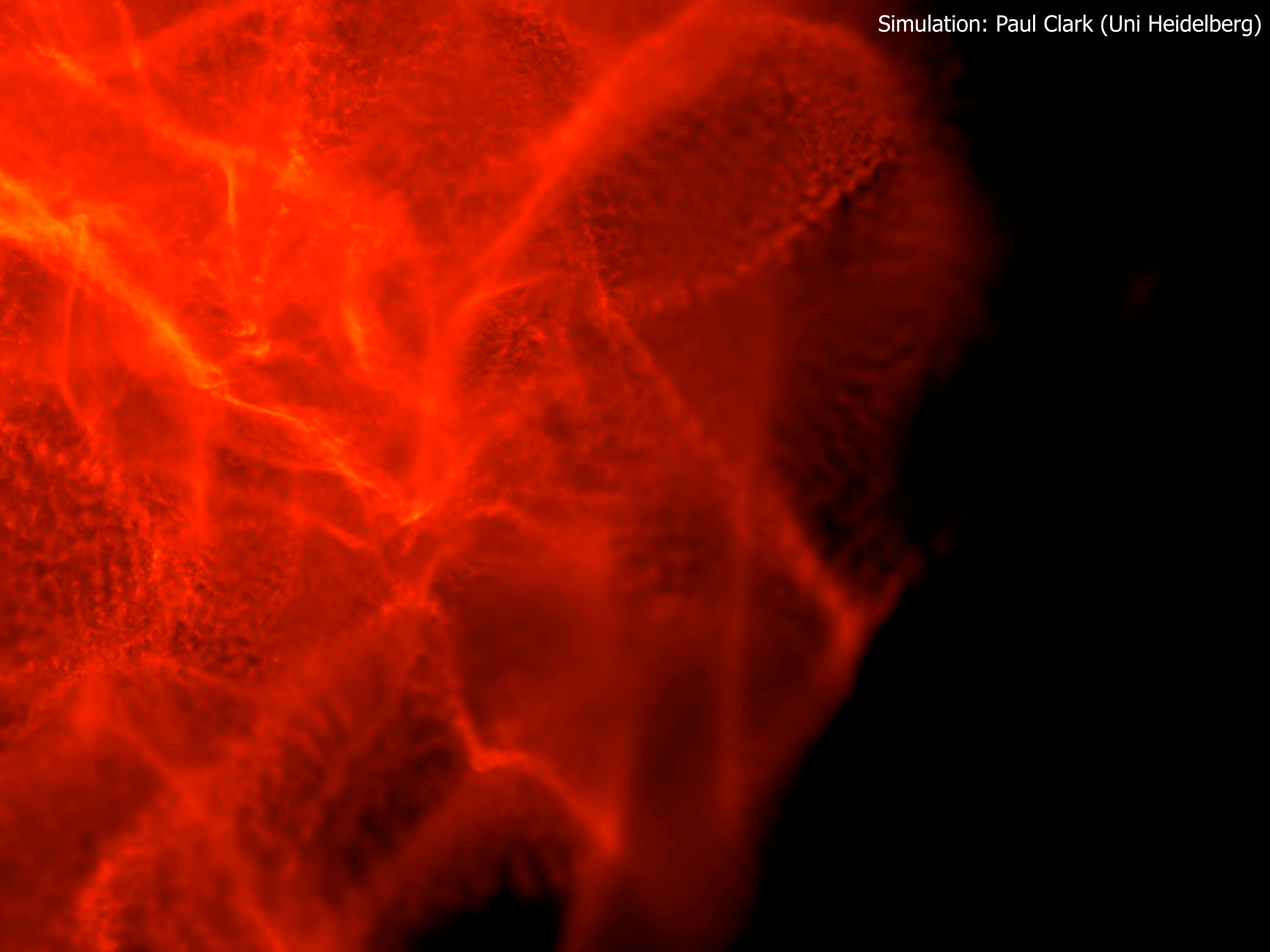
FLASH code: Sinks

BUT: sinks are a numerical 'extension' to the hydrodynamics/MHD

$\Longrightarrow$ physical interpretation ?

could be

- protostellar like objects ($r_{\mathrm{sinks}} < 0.1$ AU)
- stellar clusters ($r_{\mathrm{sinks}} \sim 1$ pc)
- binaries, little clusters? ($r_{\mathrm{sinks}} \sim 1$ AU ... 0.1 pc)


Simulation: Paul Clark (Uni Heidelberg)

FLASH code: Sinks

molecular cloud formation & evolution (*B. Körtgen*)

FLASH code: Sinks

Implementation in FLASH

- based on **`Particle`** module (*Paul Ricker*)
  - handles boundaries
  - moves particle across CPUs and blocks
  - mapping of particle properties onto grid and vice versa
  - advances particles

- Sink particle extensions (*Federrath et al.* 2010):
  - creation of particles on the fly
  - gravity: use of $1/r^2$ force for particle-particle interaction
  - time dependent mass accretion
    + linear and angular momentum transfer onto particles
  - sub-cycling of particle timestep
  - interface for back-reaction onto grid $\Longrightarrow$ feedback
  - MPI communication of **global** particle list

FLASH code: Sinks

Gravitational interaction:

1. gas–gas (g–g)
2. gas–sinks (g–s)
3. sinks–gas (s–g)
4. sinks–sinks (s–s)

$$\mathbf{g}_{\mathrm{g-g}} = -\nabla\Phi_{\mathrm{gas}}$$

$$\mathbf{g}_{\mathrm{g-s}} = -\sum_{i}^{\text{grid}} \frac{G\,M_i}{|\mathbf{r}_i(i,j,k)|^3}\mathbf{r}_i(i,j,k)$$

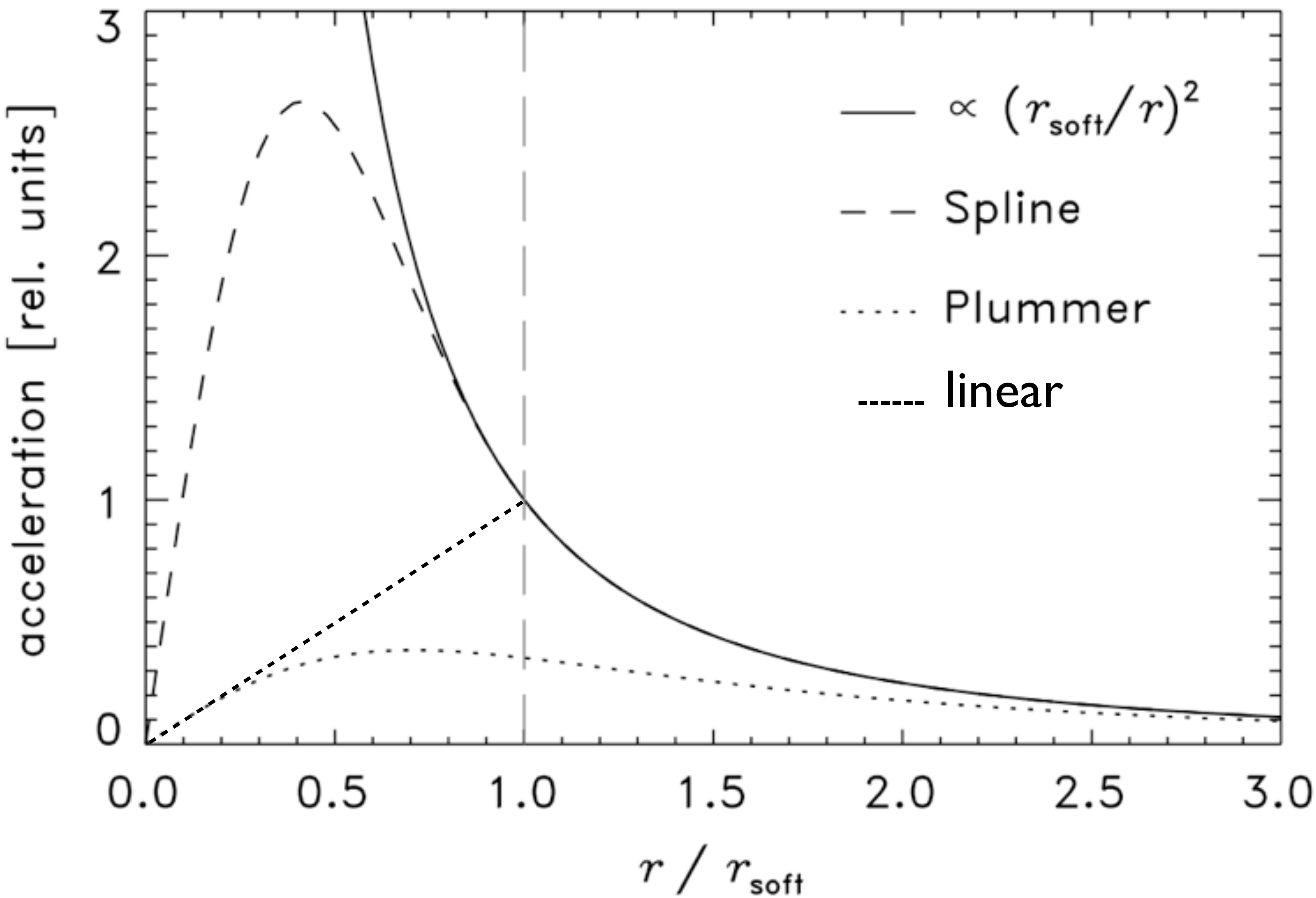
$$\mathbf{g}_{\mathrm{s-g}}(i,j,k) = -\sum_{n}^{\text{particles}} \frac{G M_n}{|\mathbf{r}_n(i,j,k)|^3}\,\mathbf{r}_n(i,j,k)$$

$$\mathbf{g}_{\mathrm{s-s},\,n} = -\sum_{m\neq n} \frac{G M_m}{|\mathbf{r}_{nm}|^3}\,\mathbf{r}_{nm}$$

with $1/r^2 \rightarrow \mathrm{f}(r, r_{\mathrm{soft}})$: gravitational softening

FLASH code: Sinks

Gravitational softening:



FLASH code: Sinks

advance particles:

- forward Euler

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^n \Delta t^n$$
$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \mathbf{a}_i^n \Delta t^n \qquad t^{n+1} = t^n + \Delta t^n$$

$\Longrightarrow$ not recommended, only for testing purposes

- variable timestep Leapfrog

$$C_n = \frac{1}{2}\Delta t^n + \frac{1}{3}\Delta t^{n-1} + \frac{1}{6}\left(\frac{\Delta t^{n2}}{\Delta t^{n-1}}\right)$$
$$D_n = \frac{1}{6}\left(\Delta t^{n-1} - \frac{\Delta t^{n2}}{\Delta t^{n-1}}\right).$$

$$\mathbf{x}_i^1 = \mathbf{x}_i^0 + \mathbf{v}_i^0 \Delta t^0$$
$$\mathbf{v}_i^{1/2} = \mathbf{v}_i^0 + \frac{1}{2}\mathbf{a}_i^0 \Delta t^0$$
$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^{n-1/2} + C_n \mathbf{a}_i^n + D_n \mathbf{a}_i^{n-1}$$
$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^{n+1/2} \Delta t^n .$$

$\Longrightarrow$ second-order time accurate

FLASH code: Sinks

Sub-cycling:

- 'close binary' interaction can limit timestep

$\Longrightarrow$ particles evolve on their own timestep

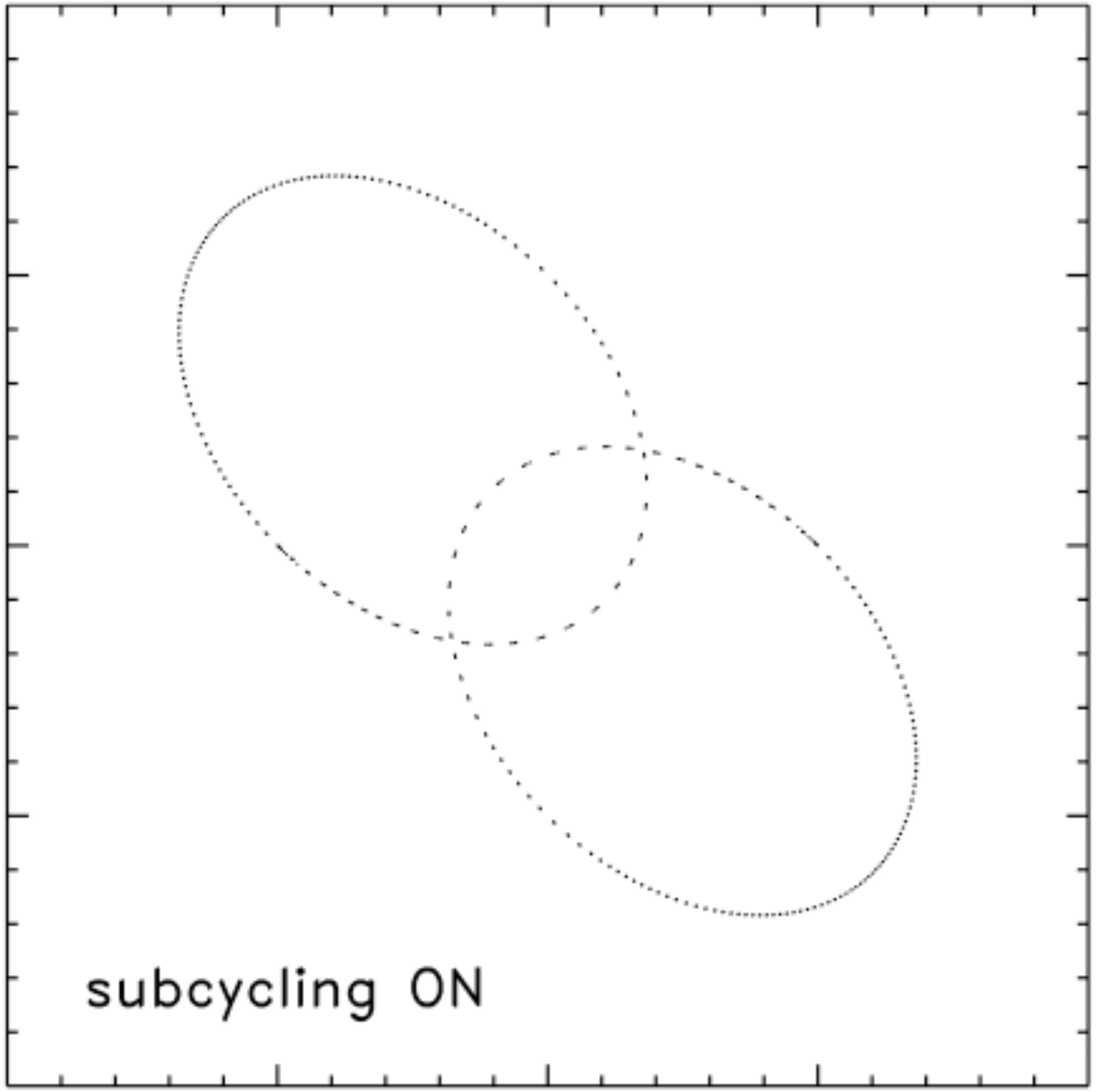
$$\Delta t_{\mathrm{gs}} = C_{\mathrm{gs}} \min_{n,m} \left(\frac{\min(|\mathbf{r}_{nm}|, \Delta x)}{|\mathbf{g}_{\mathrm{sinks},\, n}|} \right)^{1/2}$$

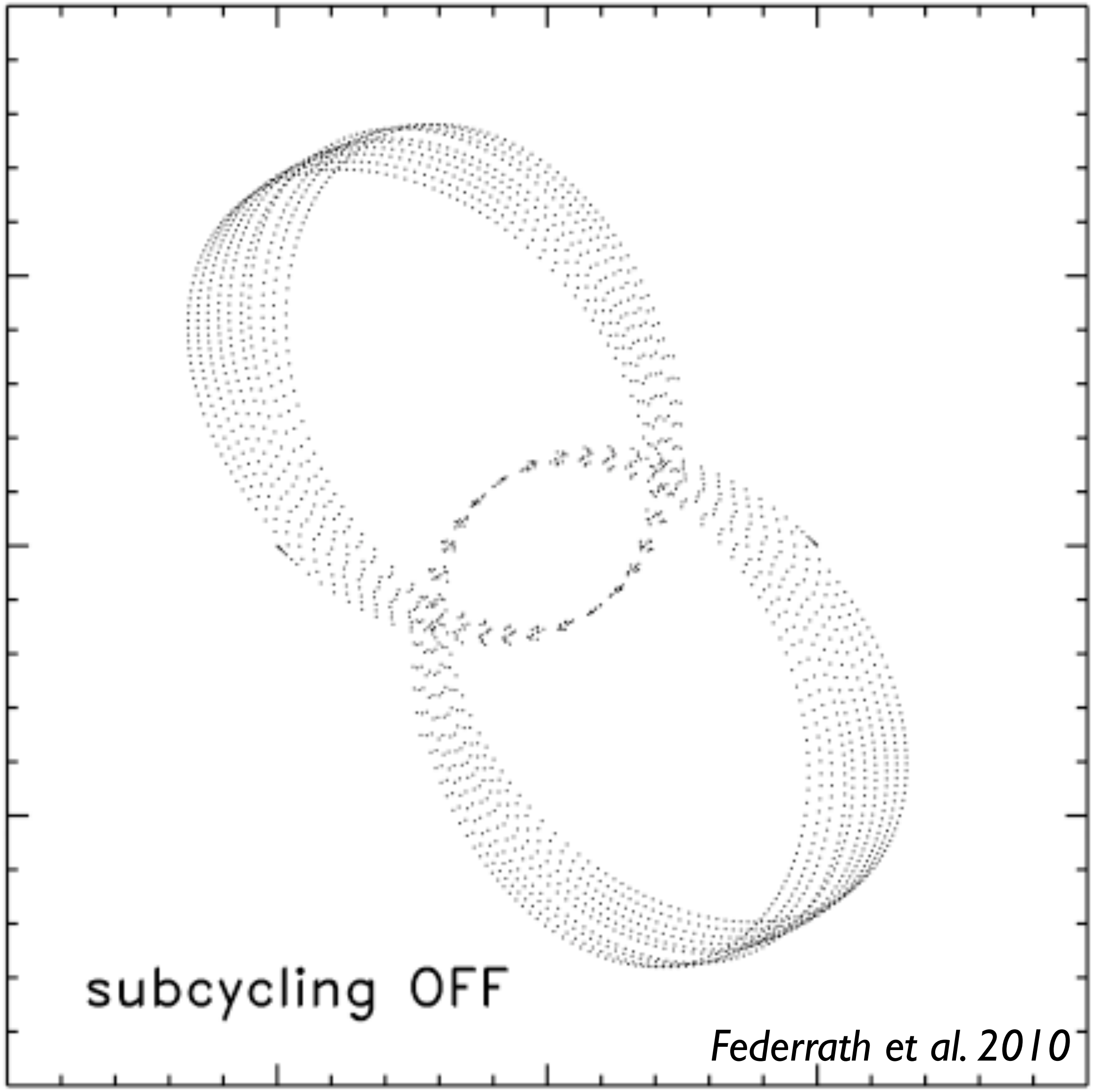
$\Longrightarrow$ use sub-cycling for particle advance:

$$N_{\mathrm{cycles}}\, \Delta t_{\mathrm{gs}} = \Delta t_{\mathrm{hydro}}$$

FLASH code: Sinks

Sub-cycling:

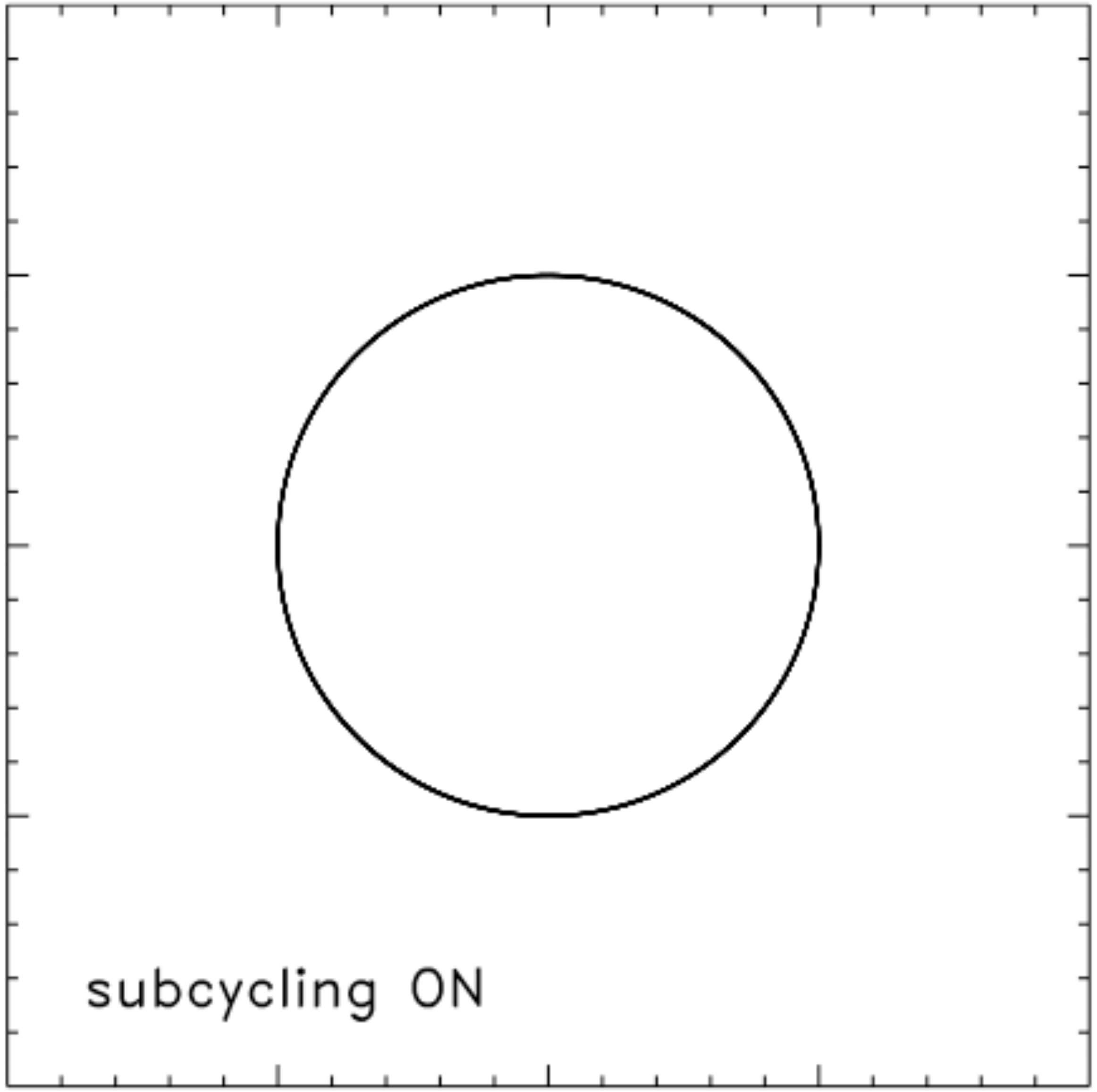




$\Longrightarrow$ after 10 orbits of two particles

FLASH code: Sinks

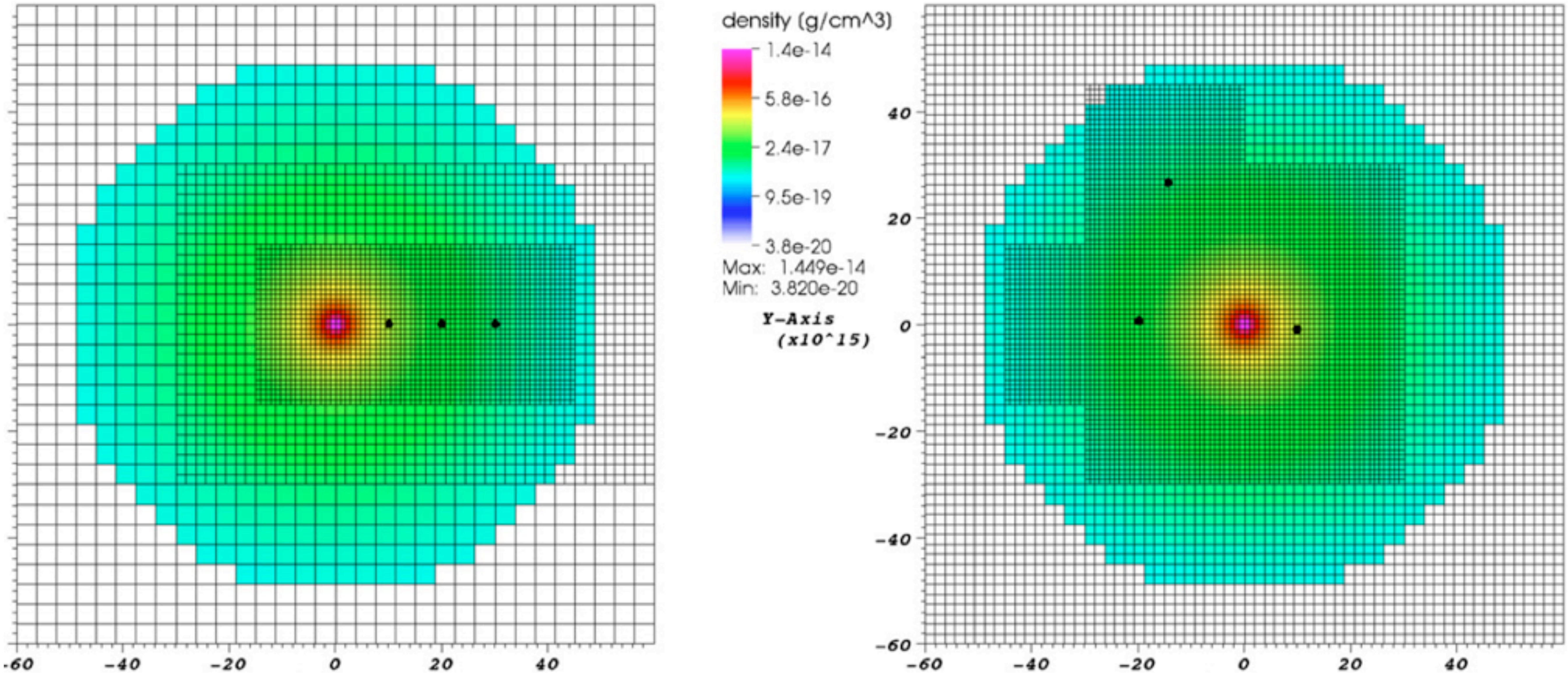
Sub-cycling:



$\Longrightarrow$ after 1000 orbits of two particles orbiting their common center

FLASH code: Sinks

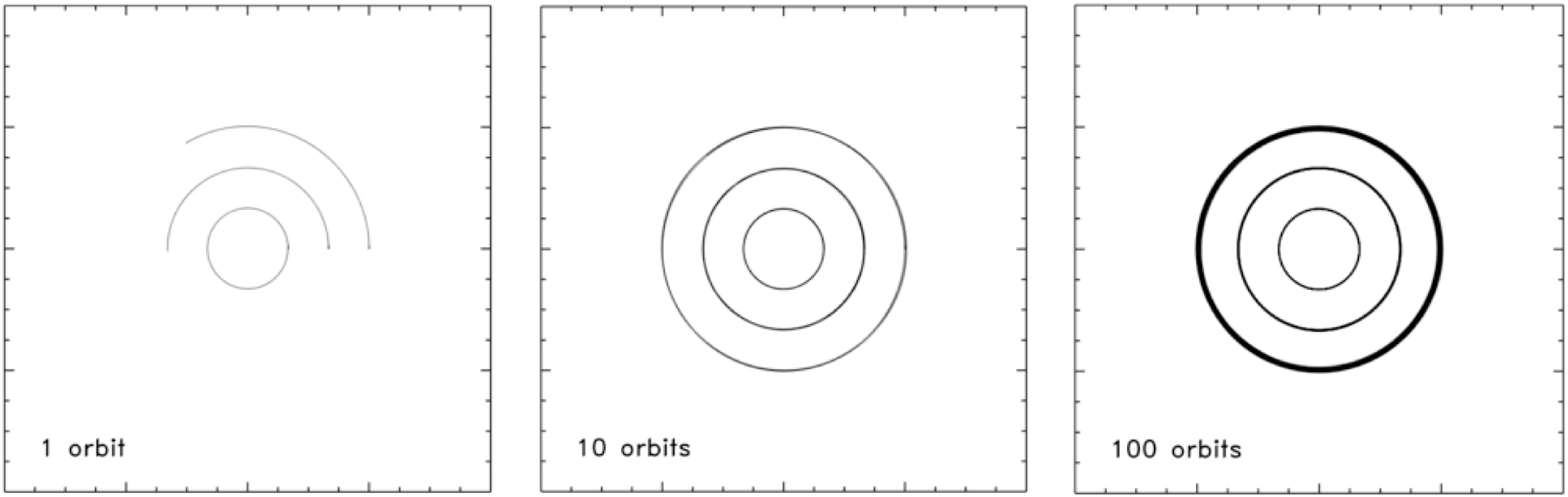
Examples / Tests: Sinks in an external potential



- dynamic sink refinement
- sub-cycling necessary

FLASH code: Sinks

Examples / Tests: Sinks in an external potential



- dynamic sink refinement
- sub-cycling necessary

FLASH code: Sinks

Create particles:

- Conditions by gravitational **collapse**:

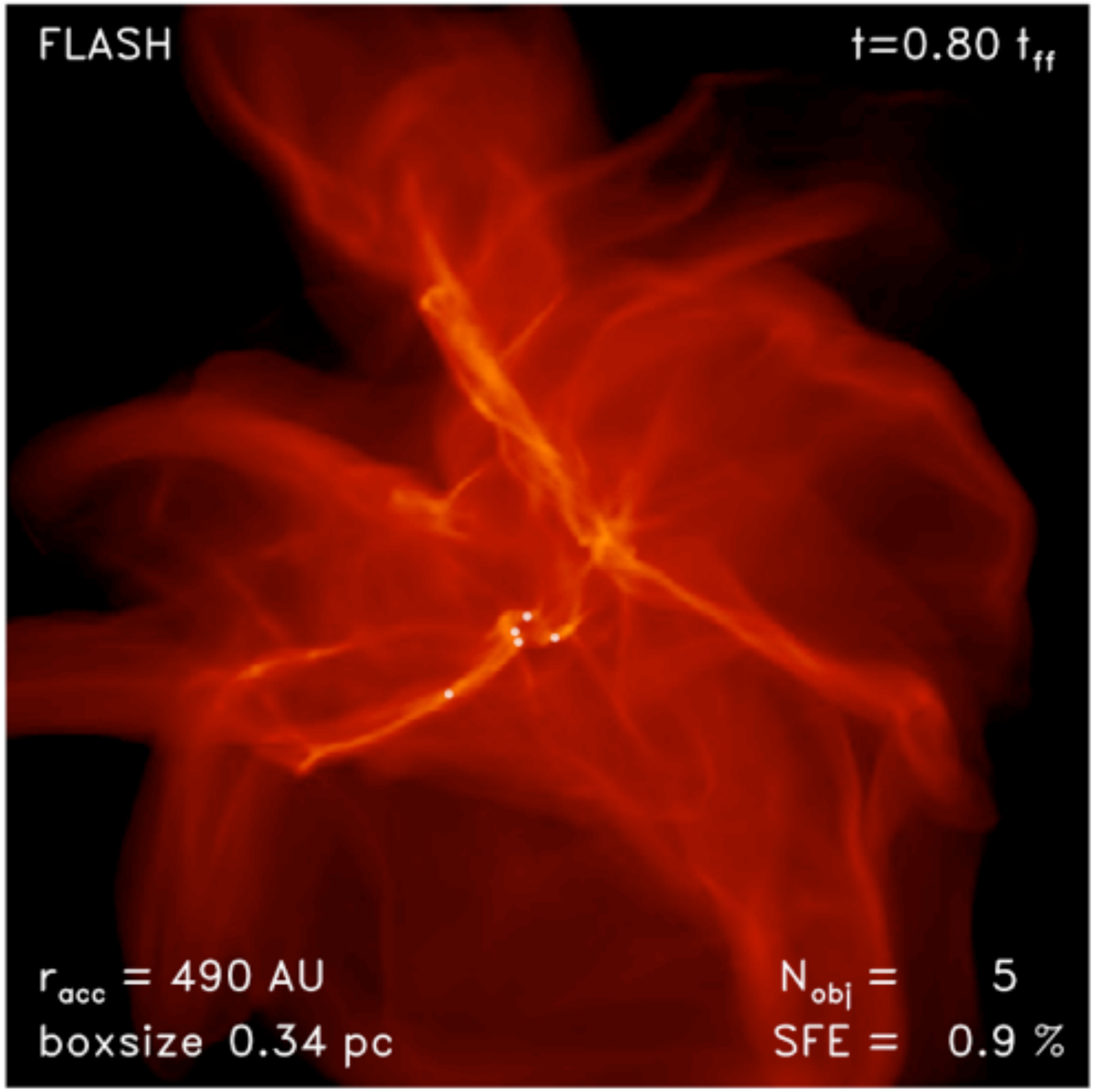
0. density criterion:

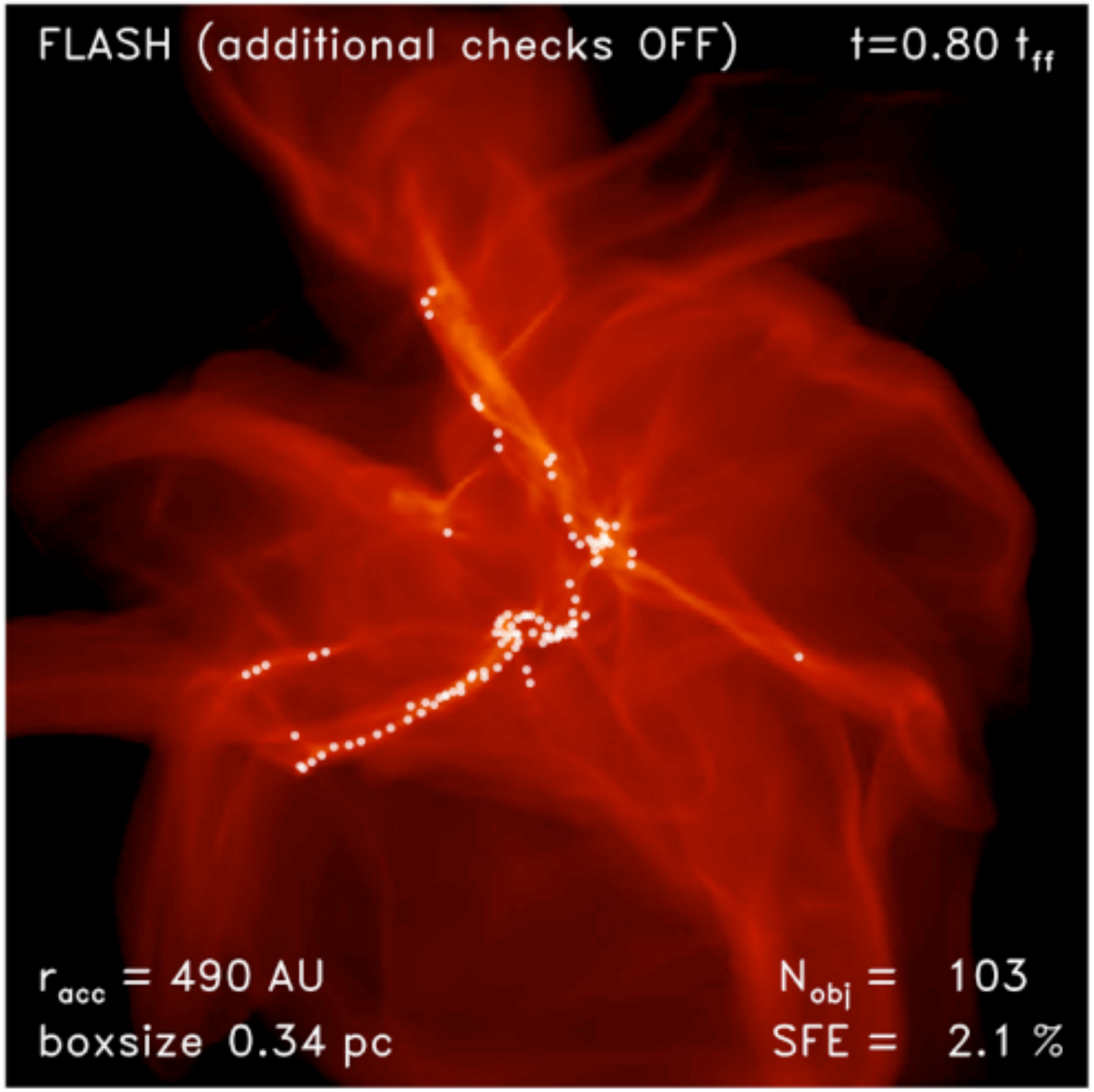
$$\rho_{\rm gas} > \rho_{\rm crit} \quad (\rho_{\rm crit}\ \text{parameter})$$

1. is on the highest level of refinement,
2. is converging, $\nabla \cdot \mathbf{v} < 0$
3. has a central gravitational potential minimum,
4. is Jeans-unstable, $|E_{\rm grav}| > 2E_{\rm th}$
5. is bound, and $E_{\rm grav} + E_{\rm th} + E_{\rm kin} + E_{\rm mag} < 0$
6. is not within $r_{\rm acc}$ of an existing sink particle.

FLASH code: Sinks

Create particles:





FLASH code: Sinks

Create particles:

- newly created particle has mass & momentum according to accretion procedure

- clear up spurious creation of sinks

  $\Longrightarrow$ only possible for very symmetric setup on multi CPUs
  e.g. spherical cloud with $r_{\rm cm}$ at block corners

  $\Longrightarrow$ merge ‘identical’ particles across different CPUs
  $\Longrightarrow$ done by master process

- source file `pt_sinkMergingAfterCreation.F90`

FLASH code: Sinks

Mass accretion & momentum transfer:

- Checks for mass accretion within $r_i < r_{\rm accr}$ (r_i distance to $r_{\rm c}$)
  - density criterion: $\rho_{\rm gas} > \rho_{\rm crit}$
  - convergent flow: $\mathrm{v}_{\rm rad} < 0$
  - gas bound: $e_{\rm kin} + e_{\rm grav} < 0$

$\Longrightarrow$ if fulfilled:

- Mass accretion from excess gas density within $r_i < r_{\rm accr}$:

$$\mathrm{M}_i = \mathrm{M}_i + \Sigma_j \, \Delta \mathrm{Vol}_j \, (\rho_j - \rho_{\rm crit})$$

- linear momentum conservation:

$$\mathrm{P}_i = \mathrm{P}_i + \Sigma_j \, \Delta \mathrm{m}_j \, \mathrm{v}_j$$

FLASH code: Sinks

angular momentum transfer?

- angular momentum conservation

$$\mathbf{R} \times \mathbf{v}_{\rm cm} = \frac{1}{M}\mathbf{L}$$

$\Longrightarrow$ **not a unique solution**

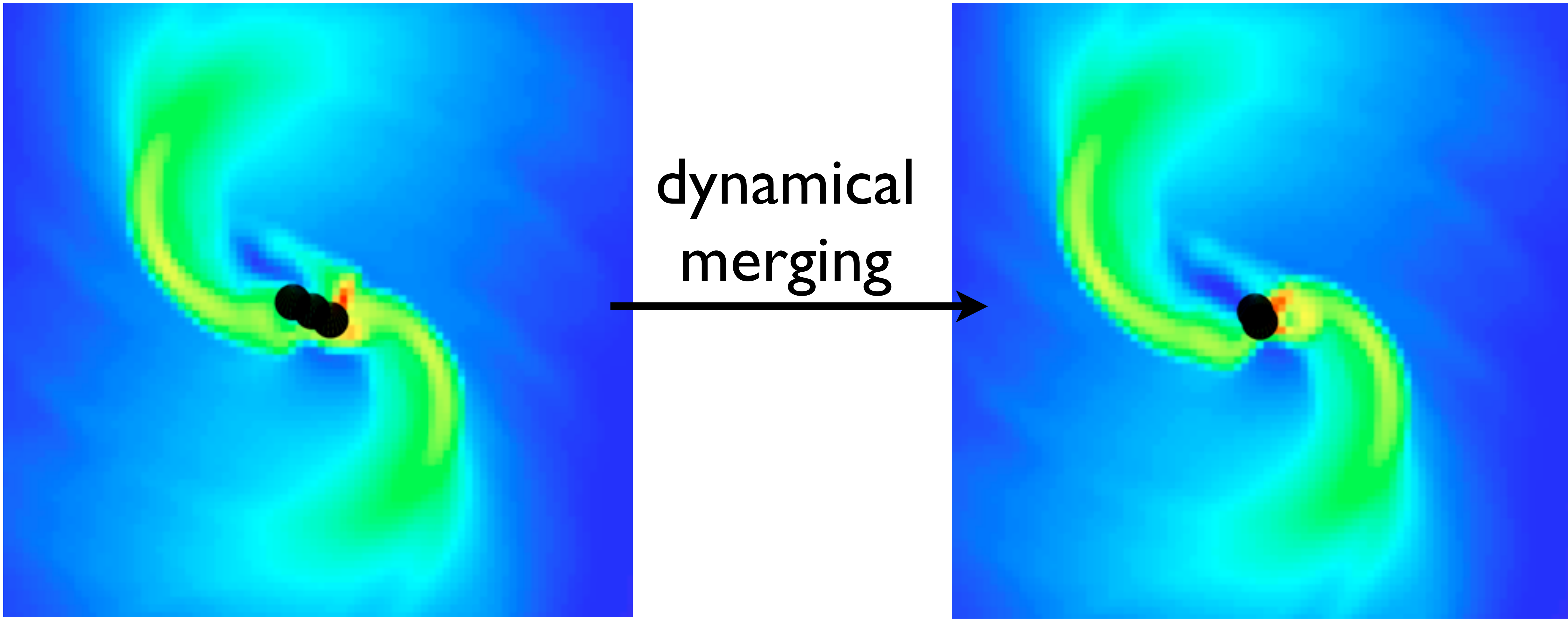
$\Longrightarrow$ use internal spin

$$\mathbf{L}_{\rm spin} = \mathbf{L}'_{\rm gas} - \mathbf{L}_{\rm gas}$$

$\Longrightarrow$ can be used for sub-grid models, i.e. for outflows properties

FLASH code: Sinks

- optional particle merging:

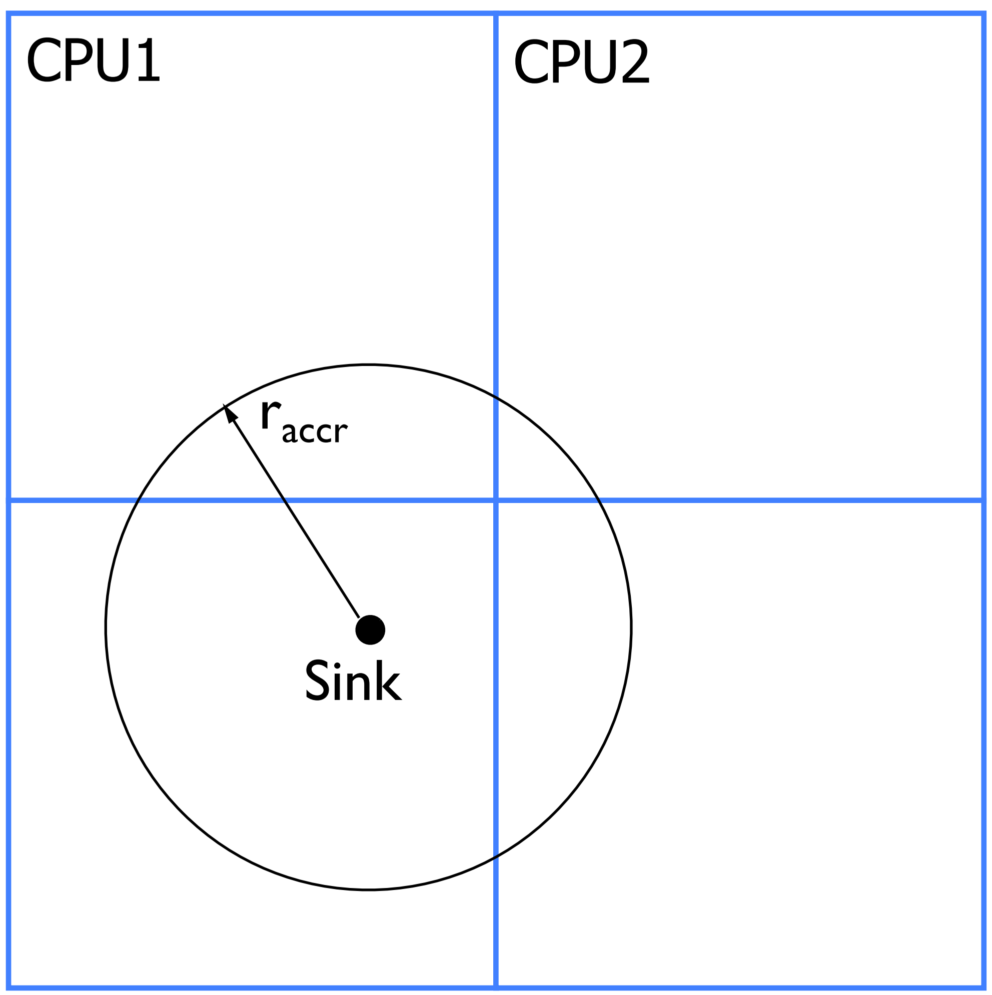


- conditions:
  - $\Longrightarrow$ particle within $r_{\rm accr}$: $r_{\rm remote} < r_{\rm accr}$
  - $\Longrightarrow$ gravitationally bound: $e_{tot} < 0$
- merge at particles center of mass
  - $\Longrightarrow$ ensures mass, momentum and angular mom. conservation

FLASH code: Sinks

- Inter CPU communication

use **global** particle list
⟹ possible for 'reasonable' number of sinks (~ 10^4)
(module: `pt_sinkGatherGlobal`)



Use global list
- to find particles
- for gas-sink interaction
- sink-sink interaction
- "integrated" quantities (e.g. total mass of all sinks)

FLASH code: Sinks

- Particle properties:

source/Particles/ParticlesMain/Config

```
PARTICLEPROP posx      REAL
PARTICLEPROP posy      REAL
PARTICLEPROP posz      REAL
PARTICLEPROP velx      REAL
PARTICLEPROP vely      REAL
PARTICLEPROP velz      REAL
PARTICLEPROP tag       REAL
PARTICLEPROP blk       REAL
PARTICLEPROP proc      REAL
```

tag: unique number

blk: block # particle resides in

proc: ProcID that holds particle

.../ParticlesMain/active/massive/Leapfrog

```
PARTICLEPROP mass     REAL        # particle mass
PARTICLEPROP accx     REAL        # x-acceleration
PARTICLEPROP accy     REAL        # y-acceleration
PARTICLEPROP accz     REAL        # z-acceleration
PARTICLEPROP oacx     REAL        # previous timestep x-acceleration
PARTICLEPROP oacy     REAL        # previous timestep y-acceleration
PARTICLEPROP oacz     REAL        # previous timestep z-acceleration
```

FLASH code: Sinks

- Particle properties:

```
.../ParticlesMain/active/Sink/Config
```

```
PARTICLEPROP accr_radius     REAL
PARTICLEPROP x_ang           REAL
PARTICLEPROP y_ang           REAL
PARTICLEPROP z_ang           REAL
PARTICLEPROP x_ang_old       REAL
PARTICLEPROP y_ang_old       REAL
PARTICLEPROP z_ang_old       REAL
PARTICLEPROP accr_rate       REAL
PARTICLEPROP old_pmass       REAL
PARTICLEPROP creation_time   REAL
PARTICLEPROP tag             REAL
PARTICLEPROP dtold           REAL
PARTICLEPROP mgas            REAL
PARTICLEPROP x_bflux         REAL
PARTICLEPROP y_bflux         REAL
PARTICLEPROP z_bflux         REAL
```

} not used: what to do with B-fields?

FLASH code: Sinks

- Parameters:

`.../ParticlesMain/active/Sink/Config`

```
PARAMETER sink_density_thresh        REAL     1.0e-14
PARAMETER sink_accretion_radius      REAL     1.0e14
PARAMETER sink_softening_radius      REAL     1.0e14
PARAMETER sink_softening_type_gas    STRING   "linear"
PARAMETER sink_softening_type_sinks  STRING   "spline"
PARAMETER sink_integrator            STRING   "leapfrog"
PARAMETER sink_subdt_factor          REAL     0.01
PARAMETER sink_dt_factor             REAL     0.5
PARAMETER sink_merging               BOOLEAN  FALSE
```

$\Longrightarrow$ choose ρ_{thres} so that Truelove criterion is not violated:

$$\lambda_J > N_J \, \Delta x_{\min} \quad ; \quad N_J > 4$$

with $\lambda_J = (\pi c^2/G\rho)^{1/2}$ and $\Delta x_{\min} = L_{\mathrm{box}}/2^{l\max+2}$

$\Longrightarrow \rho_{\mathrm{thres}}(l_{\max}) \approx 4^{l\max+2} \, \pi c^2/(G \, N_J^2 \, L_{\mathrm{box}}^2)$

FLASH code: Sinks

- Jeans refinement criterion:
  `.../ParticlesMain/active/Sink/Config`

```
# Refinement on Jeans length and sink particles
PARAMETER refineOnJeansLength     BOOLEAN  TRUE
PARAMETER refineOnSinkParticles   BOOLEAN  TRUE
PARAMETER jeans_ncells_ref        REAL     32.0
PARAMETER jeans_ncells_deref      REAL     64.0
```

$\Longrightarrow$ refine if

$$\Delta x < \texttt{jeans\_ncells\_ref} \times \lambda_J(\rho(x))$$

$\Longrightarrow$ de-refine if

$$\Delta x > \texttt{jeans\_ncells\_deref} \times \lambda_J(\rho(x))$$

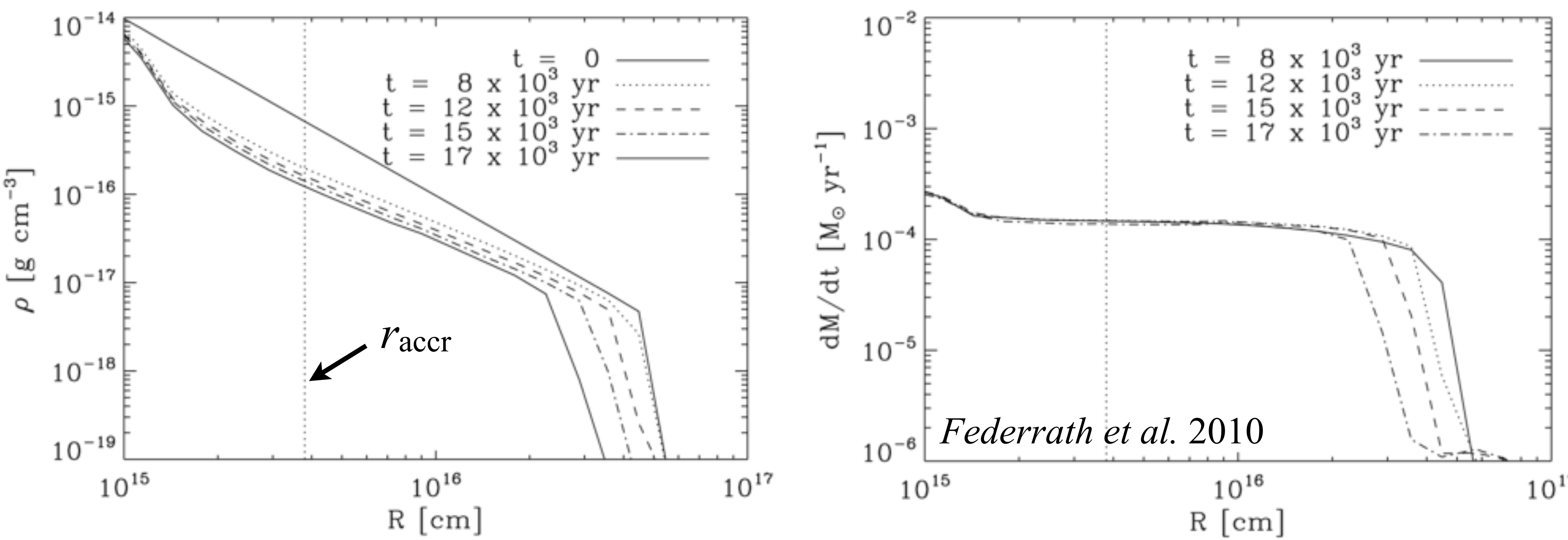
FLASH code: Sinks

Examples / Tests: Collapse of a singular-isothermal-sphere

(SIS, *Shu* 1977)

$\Longrightarrow$ collapse of a static sphere with

$\rho(r) \propto r^{-2}$

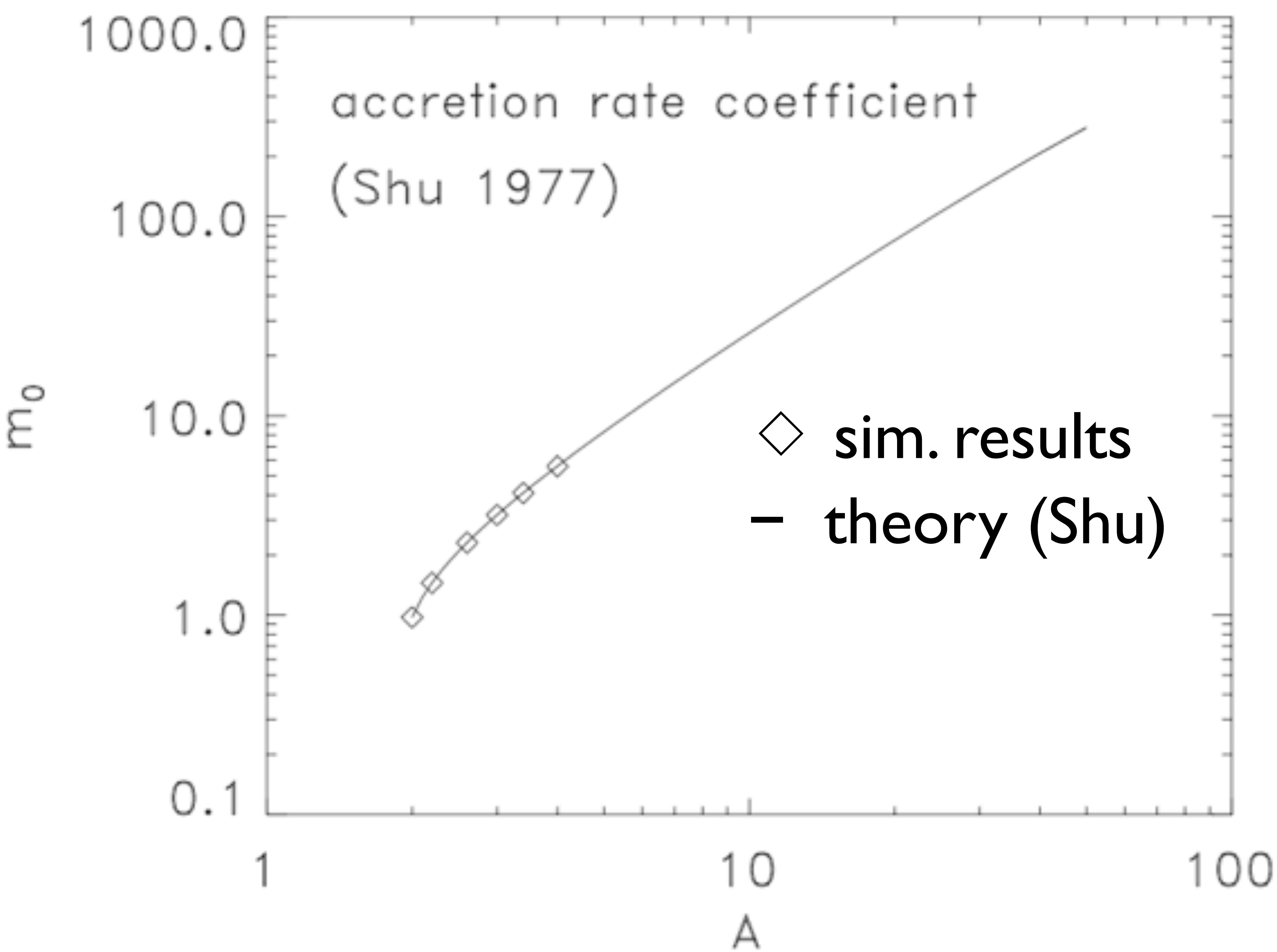


$\Longrightarrow$ constant mass accretion rate: $\propto c^3/G_N$

FLASH code: Sinks

Examples / Tests: Collapse of a singular-isothermal-sphere

(SIS, *Shu* 1977)



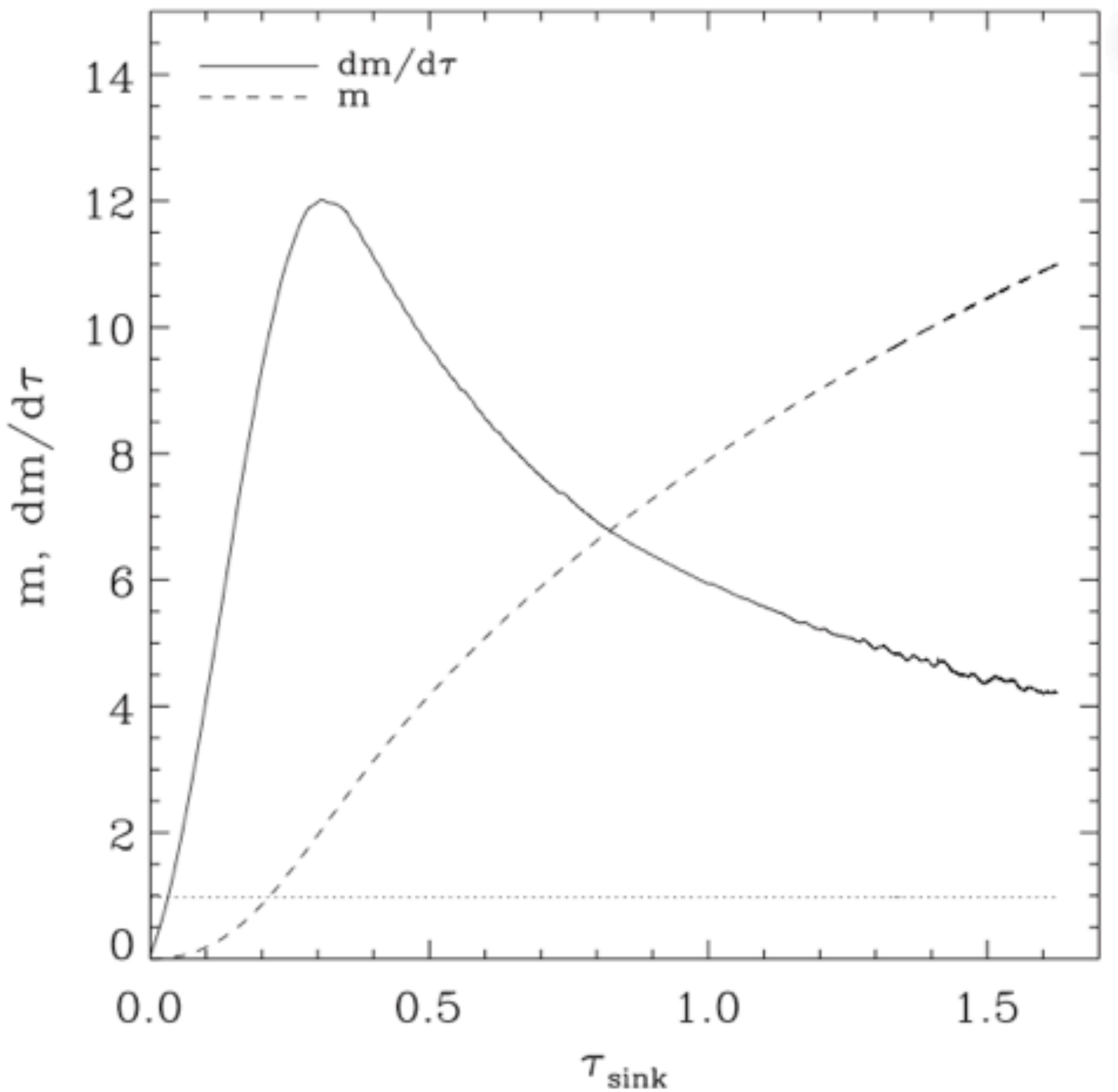
$\Longrightarrow$ constant mass accretion rate for a given core mass M_{core}:

$$\dot{M} = m_0 \frac{c_s^3}{G}$$

A: instability parameter $A(M_{\mathrm{core}})$

FLASH code: Sinks

Examples / Tests: Collapse of a Bonnor-Ebert sphere



$$\xi = \frac{r}{c_s/\sqrt{4\pi\, G\, \rho_0}}$$

$$\tau = \frac{t}{1/\sqrt{4\pi\, G\, \rho_0}}$$

$$m = \frac{M}{c_s^3/\sqrt{4\pi\, G^3\, \rho_0}}$$

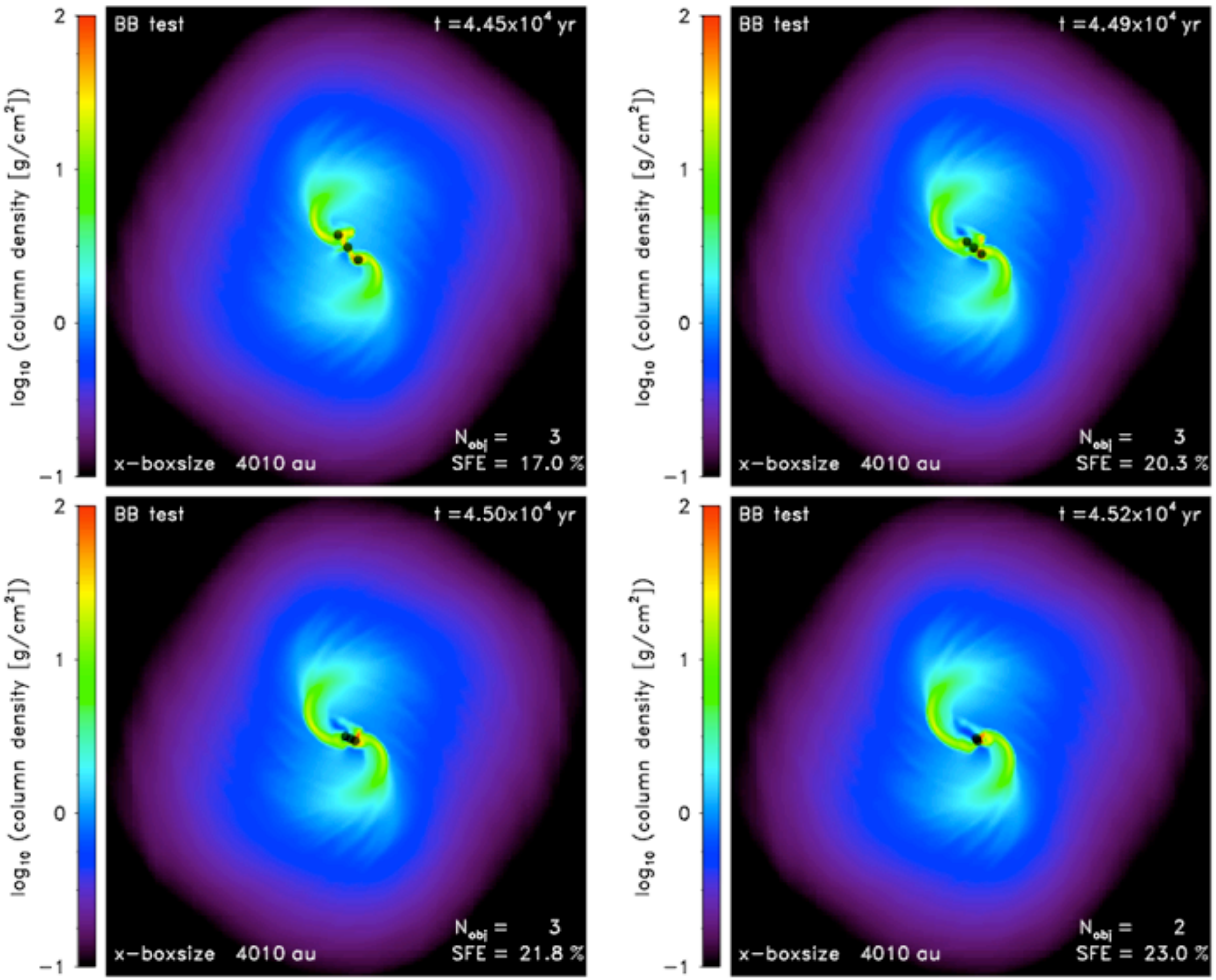
$$\dot{m} = \frac{\dot{M}}{c_s^3/G}\,,$$

$\Longrightarrow$ no analytic solution, but good agreement with *Foster & Chevalier* 1993

FLASH code: Sinks

Examples / Tests: Boss-Bodenheimer Test

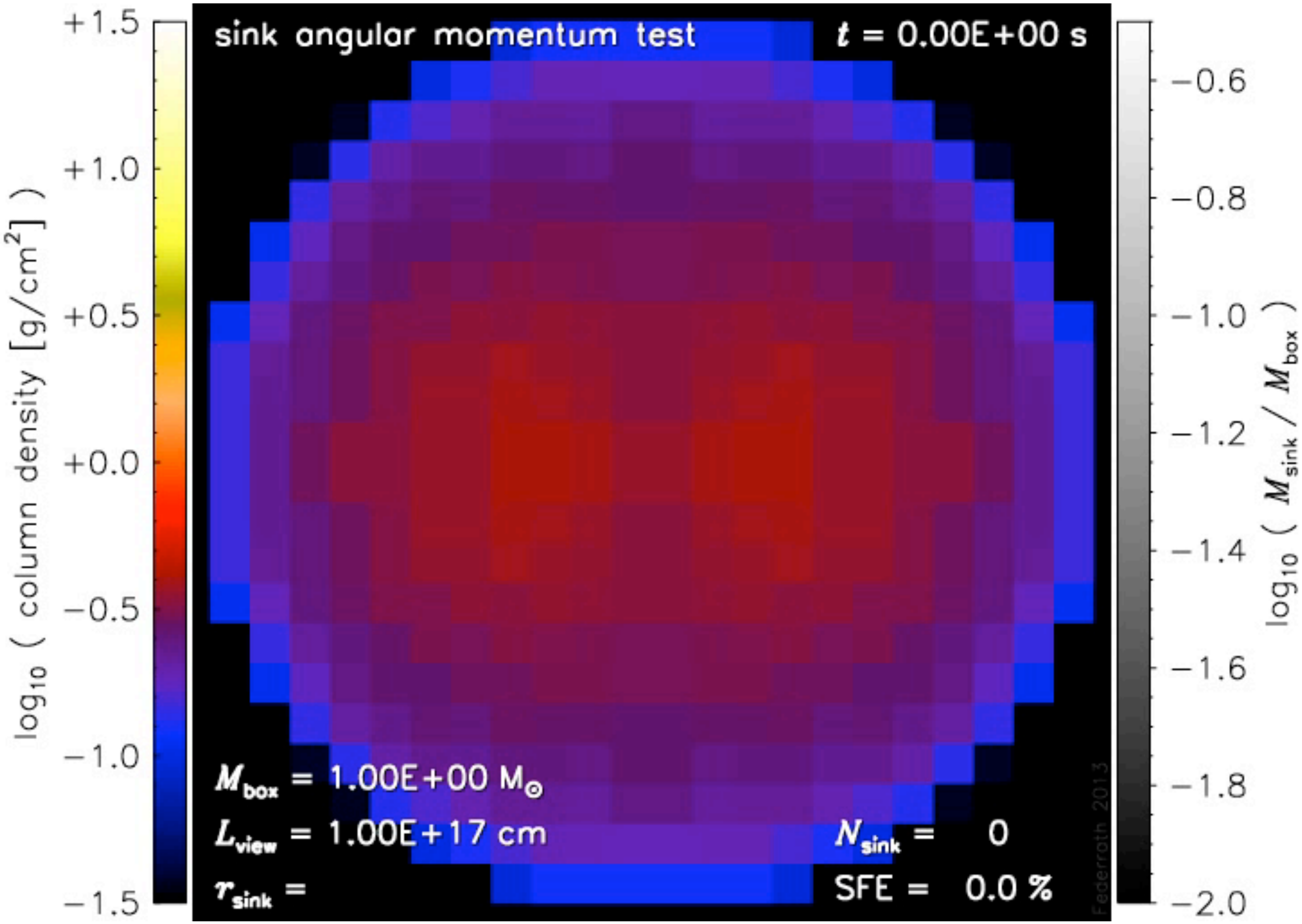
(*Boss & Bodenheimer* 1979)



FLASH code: Sinks

Examples / Tests: Boss-Bodenheimer Test

(*Boss & Bodenheimer* 1979)



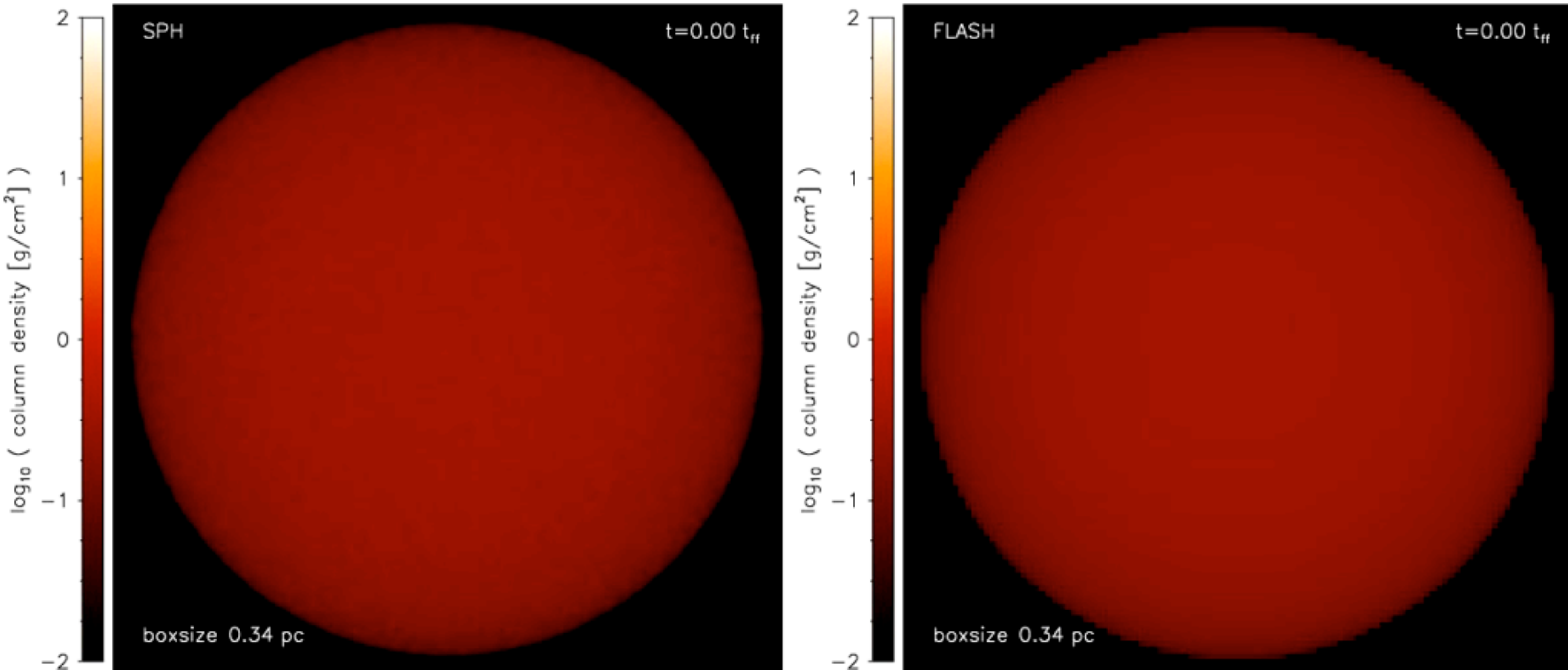
FLASH code: Sinks

Examples / Tests: Collapse of a turbulent cloud core

$\Longrightarrow$ comparison with SPH simulations (Paul Clark)

FLASH code: Sinks

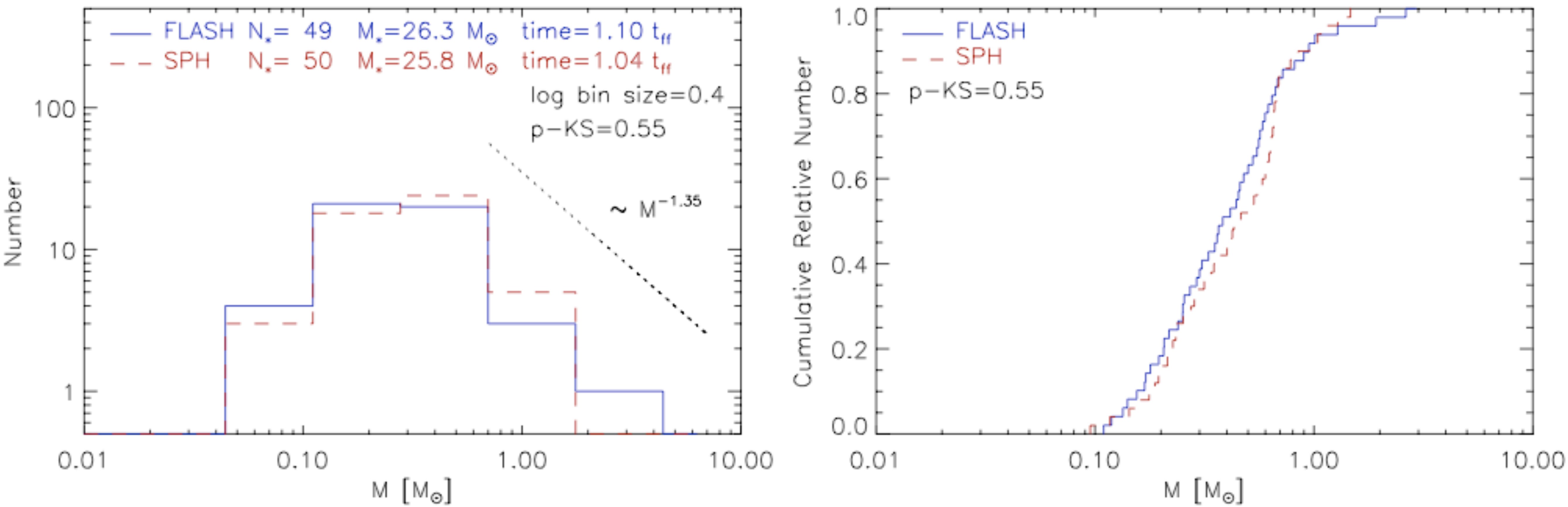
Examples / Tests: Collapse of a turbulent cloud core



⟹ comparison with SPH simulations (Paul Clark)

FLASH code: Sinks

Examples / Tests: Collapse of a turbulent cloud core



- good agreement
- differences due to hydro

  ⟹ SPH slightly more dissipative

  ⟹ collapses slightly faster

  ⟹ cluster more centrally condensed

FLASH code: Sinks

Examples:

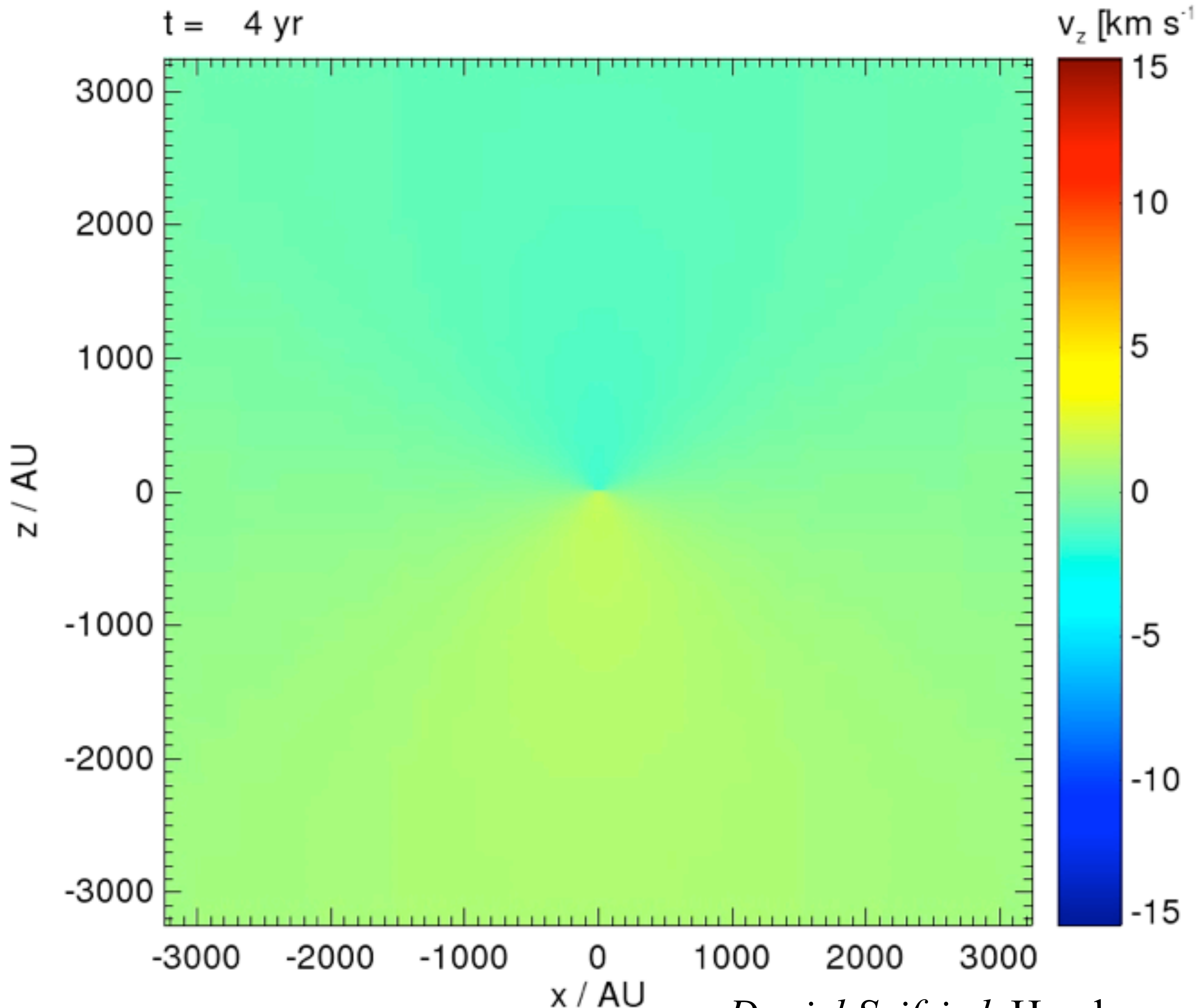
long-term evolution of a magnetised collapsing cloud core

$\Longrightarrow$ self-consistent launch of an outflow

FLASH code: Sinks

Examples:
long-term evolution of a magnetised collapsing cloud core

⟹ self-consistent launch of an outflow

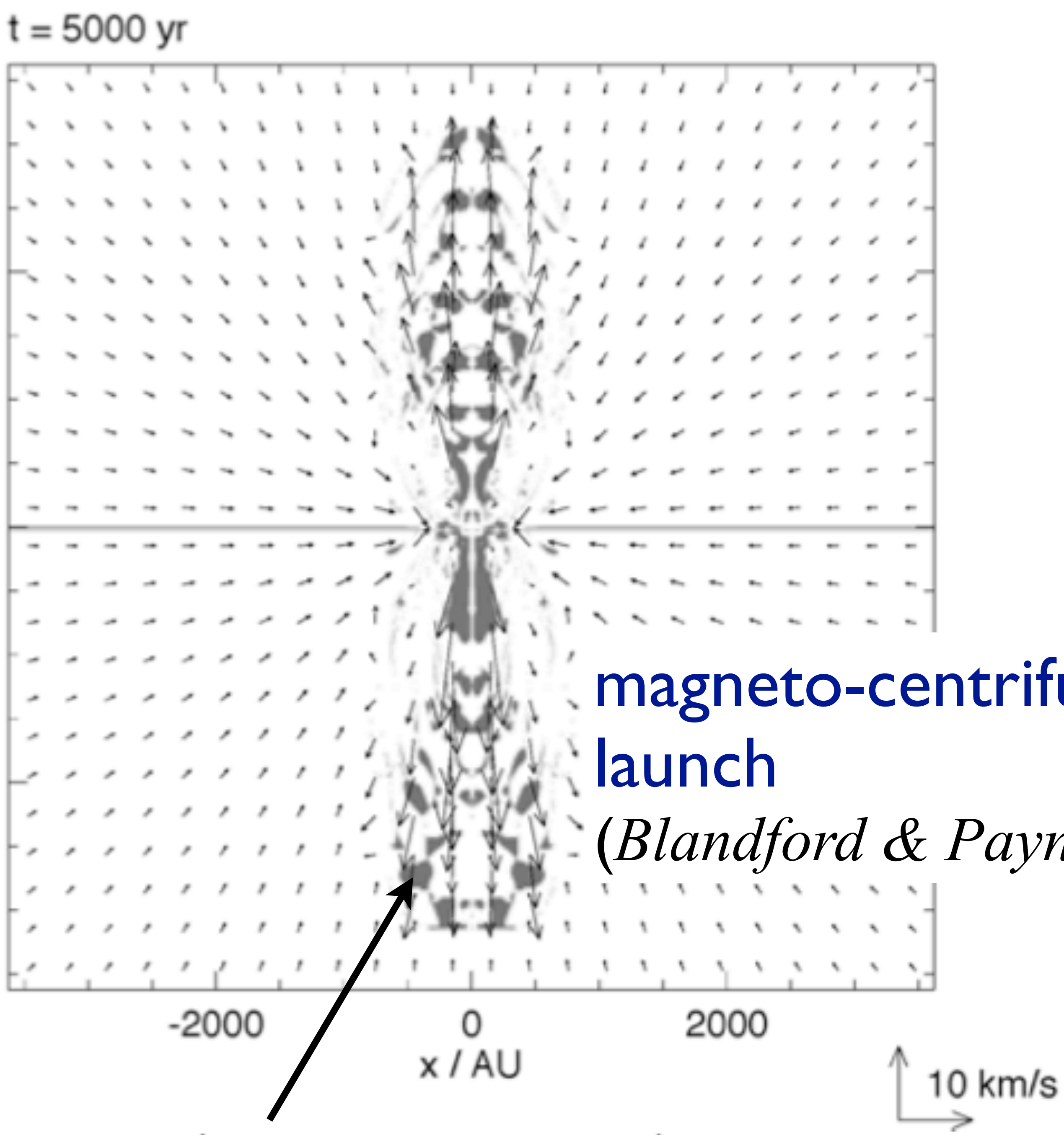


Daniel Seifried, Hamburg

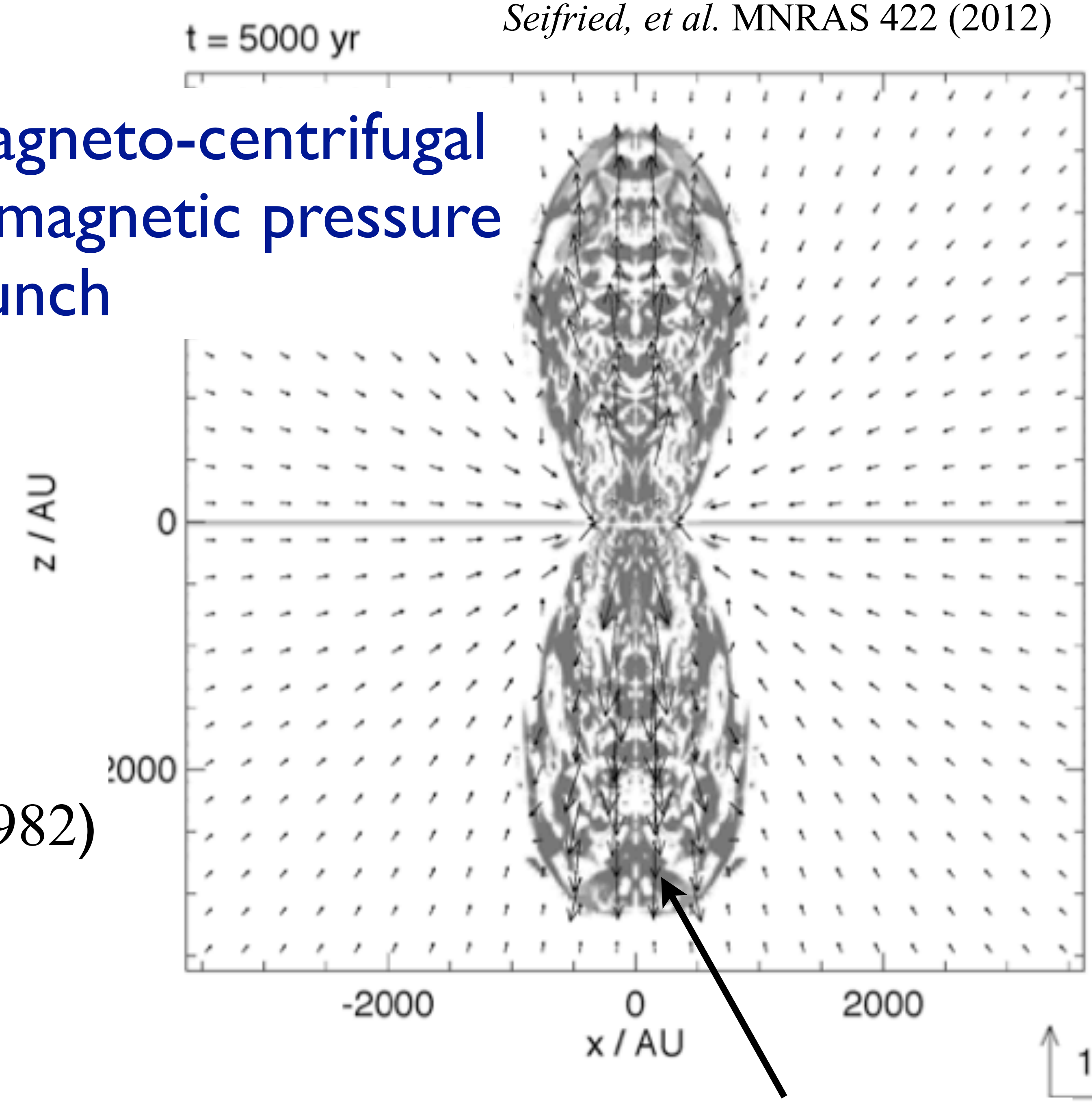
FLASH code: Sinks

Examples: $\Longrightarrow$ self-consistent launch of an outflow

Seifried, et al. MNRAS 422 (2012)



magneto-centrifugal
launch
(*Blandford & Payne* 1982)



magneto-centrifugal
+ magnetic pressure
launch

$$\frac{r}{z}\frac{1}{GM}\left(\frac{v_\phi^2}{r^2}(r^2+z^2)^{3/2} - GM\right) \Big/ \left(\frac{B_z}{B_r}\right) > 1$$

$$\partial_{\rm pol}\left(\frac{1}{2}v_\phi^2 + \Phi - \frac{v_\phi}{v_{\rm pol}}\frac{1}{4\pi}\frac{B_\phi B_{\rm pol}}{\rho} + \frac{1}{4\pi}\frac{B_\phi^2}{\rho}\right) < 0$$

FLASH code: Sinks

Examples: sub-grid outflow feedback

modelled on sinks

$\Longrightarrow$ momentum injection
$\propto \mathrm{d}M_{\mathrm{accr}}/\mathrm{d}t$

$\Longrightarrow$ const opening angle
$\sim 30^\circ$

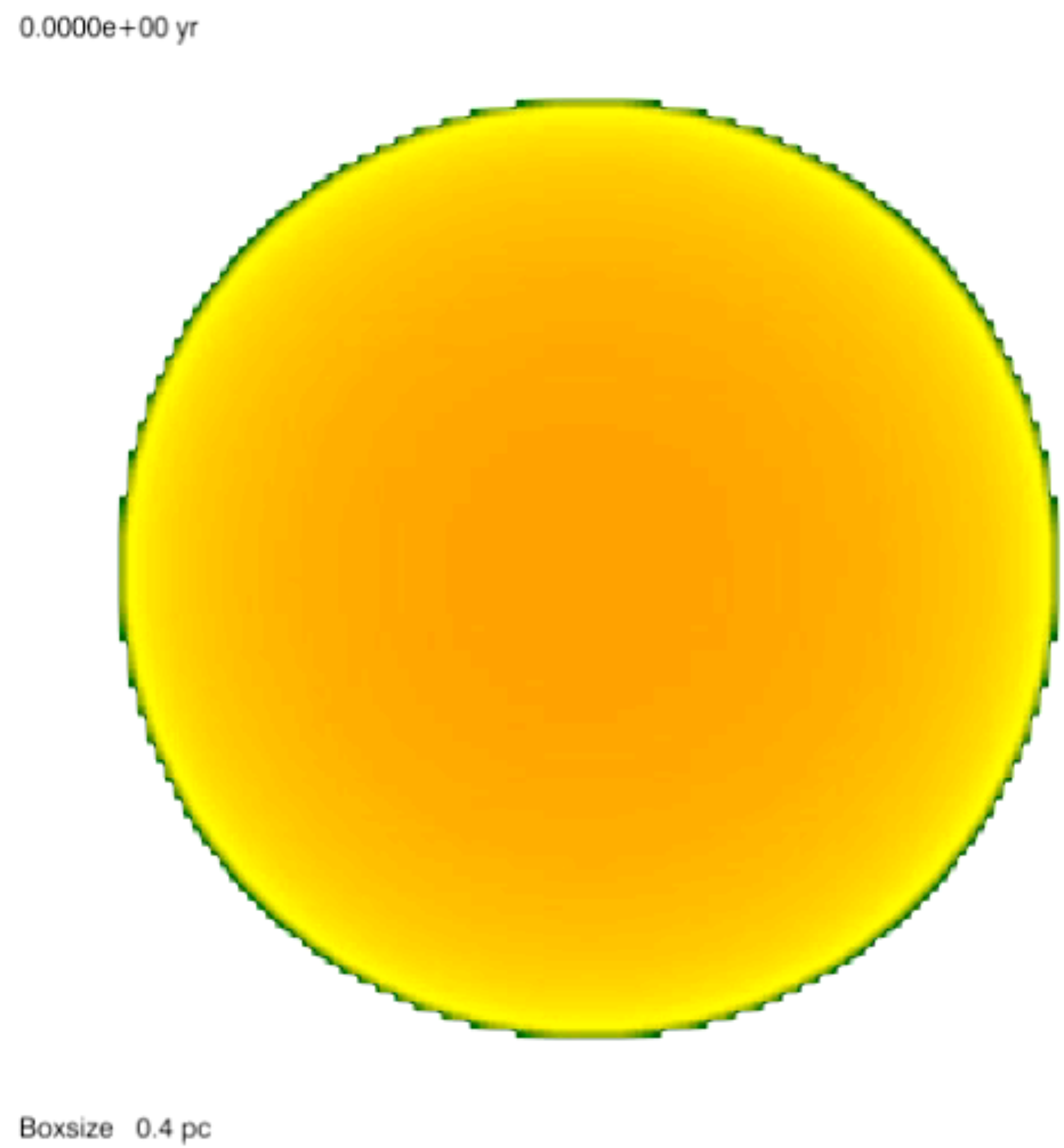
FLASH code: Sinks

Examples: sub-grid outflow feedback

modelled on sinks

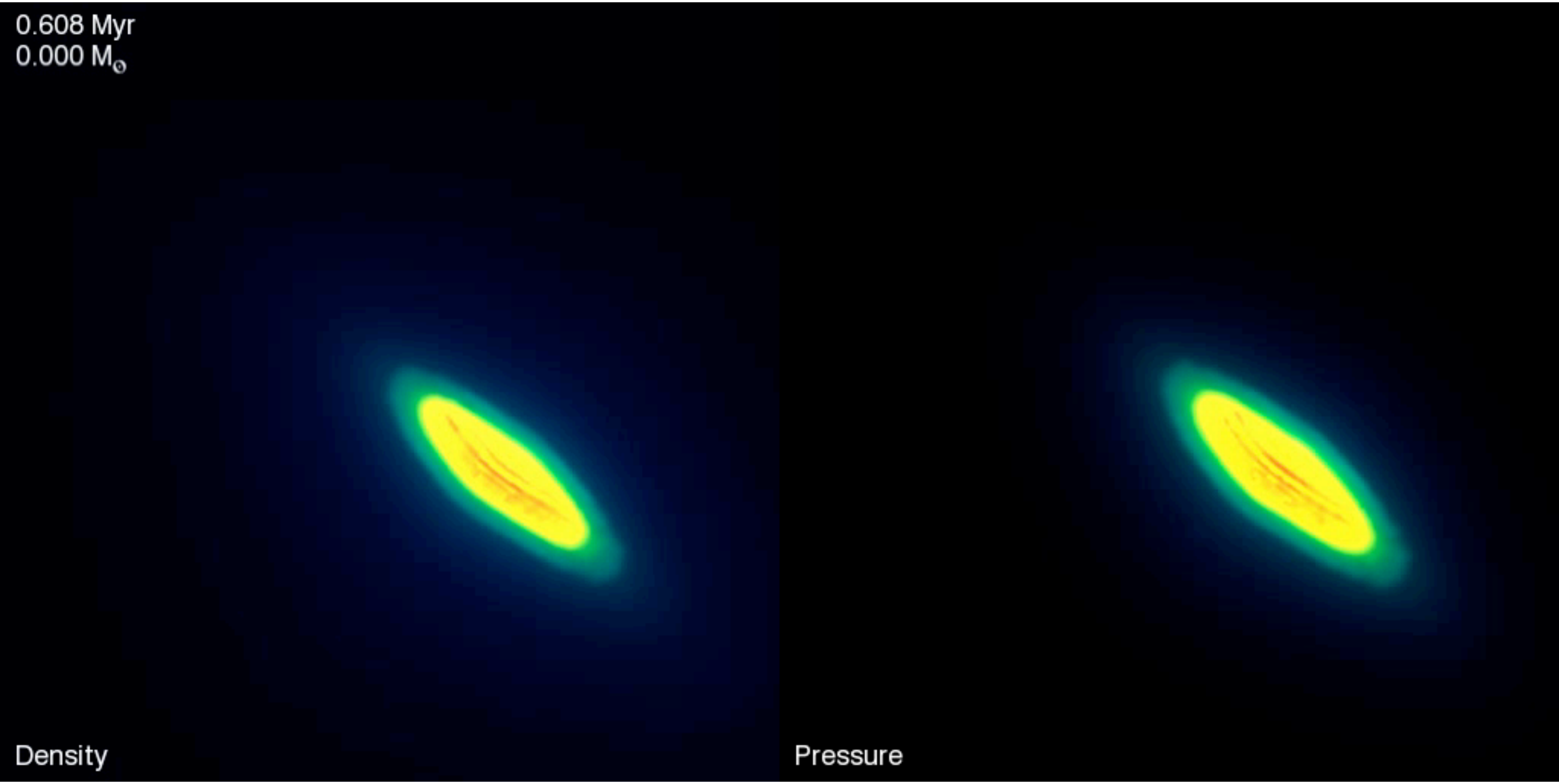
$\Longrightarrow$ momentum injection
$\propto \mathrm{d}M_{\mathrm{accr}}/\mathrm{d}t$

$\Longrightarrow$ const opening angle
$\sim 30°$



FLASH code: Sinks

Examples: feedback from ionising radiation by massive stars



Peters et al. 2010

courtesy: Zilken, NIC, Jülich

FLASH code: Sinks

Examples: sub-grid supernova feedback

⟹ kinetic and thermal energy injection

$$E_{\mathrm{tot}} \sim 10^{51}\ \mathrm{erg}$$

(*Bastian Körtgen*)

FLASH code: Sinks

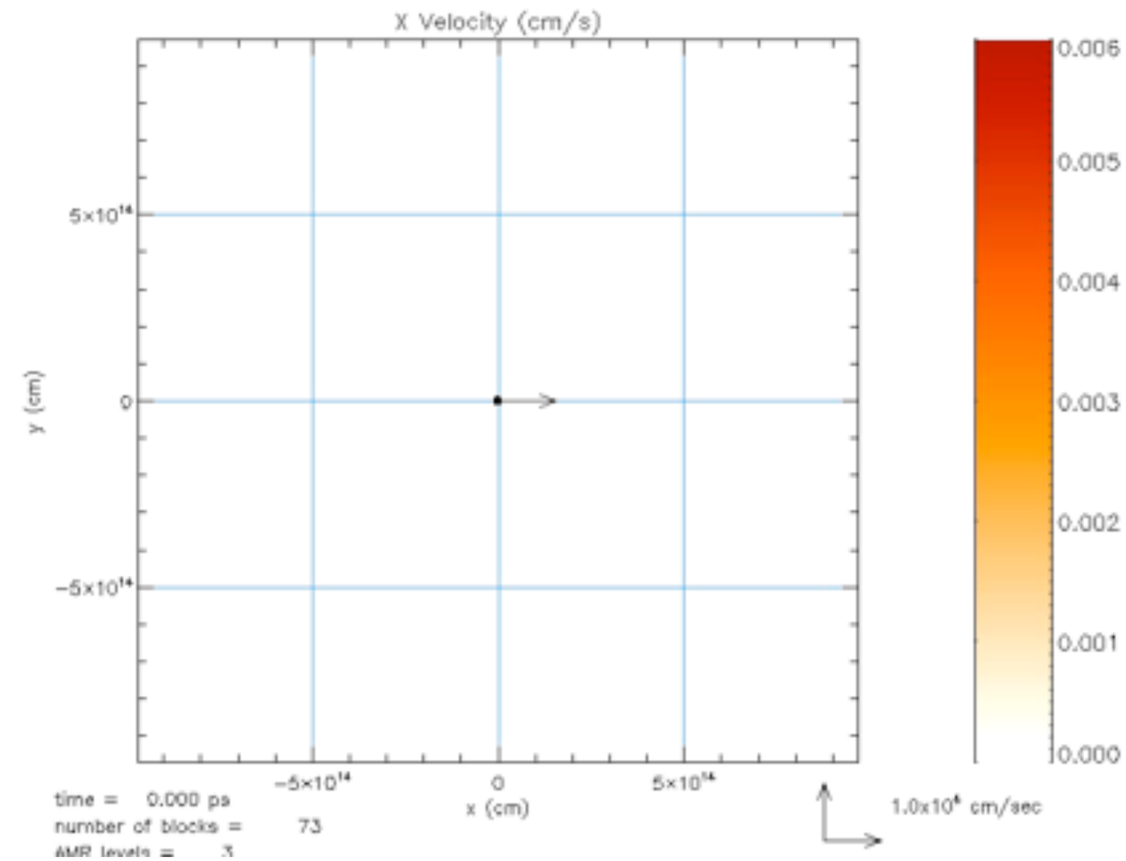
Examples: gas drag on particles

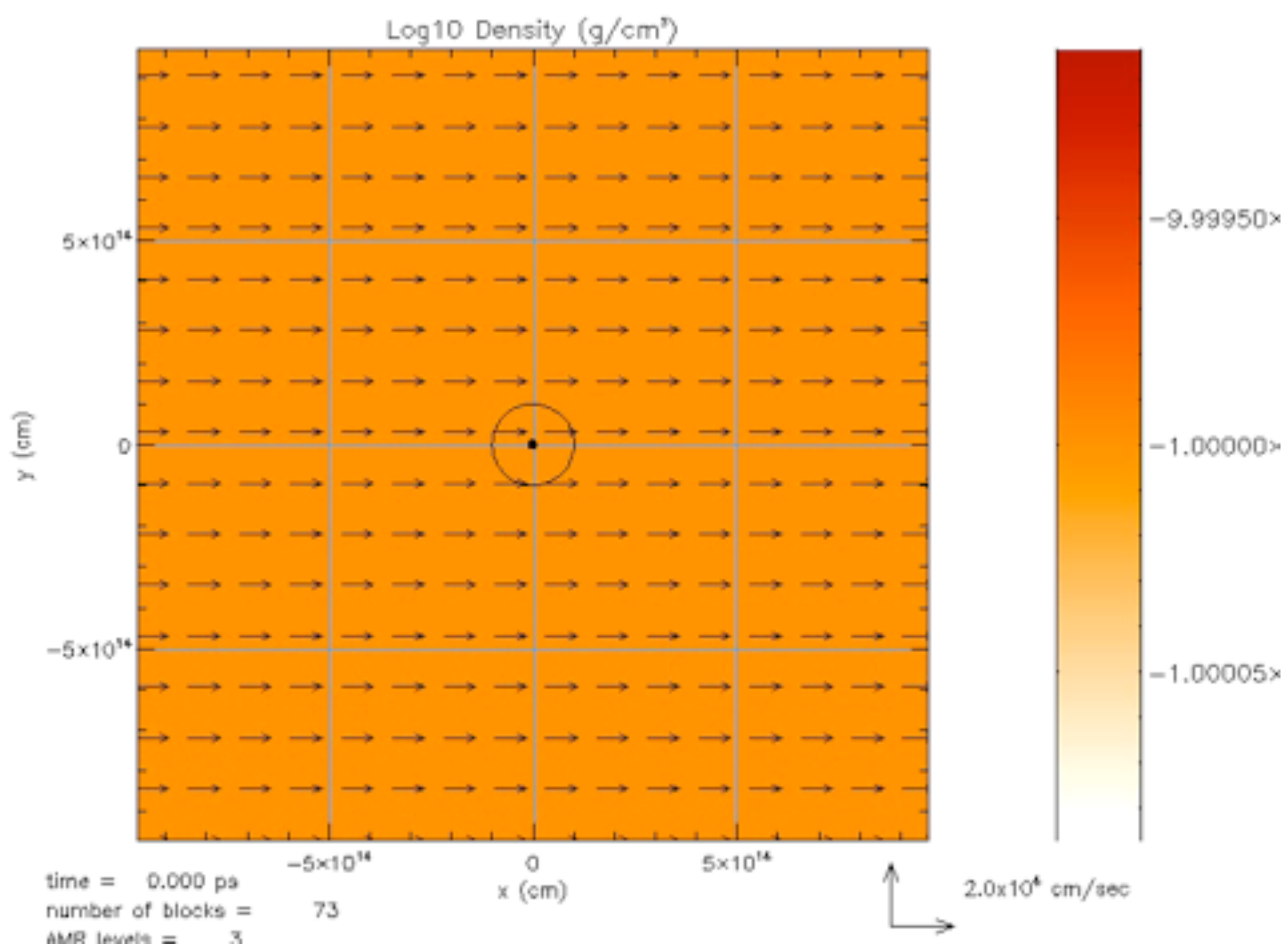
- “easy” implementations of gas-particle interaction
- e.g. particle-gas drag via elastic scattering

FLASH code: Sinks

Examples: gas drag on particles

- "easy" implementations of gas-particle interaction
- e.g. particle-gas drag via elastic scattering

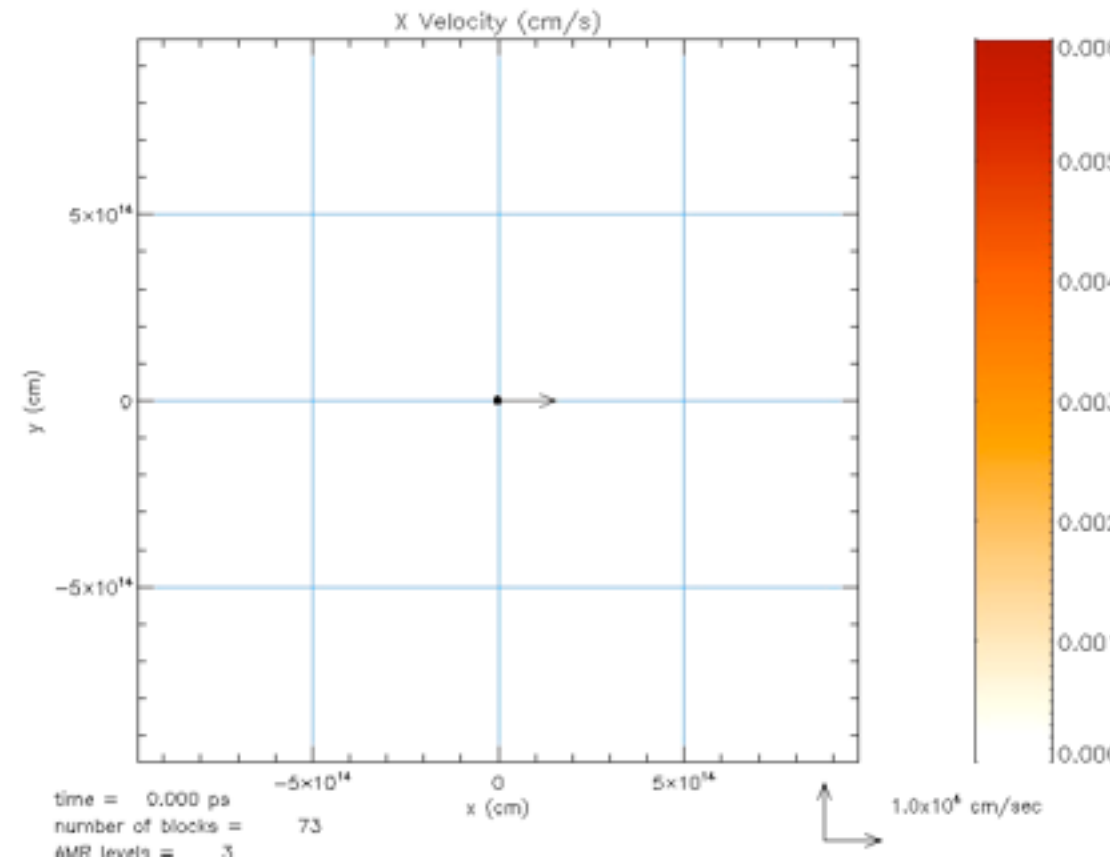


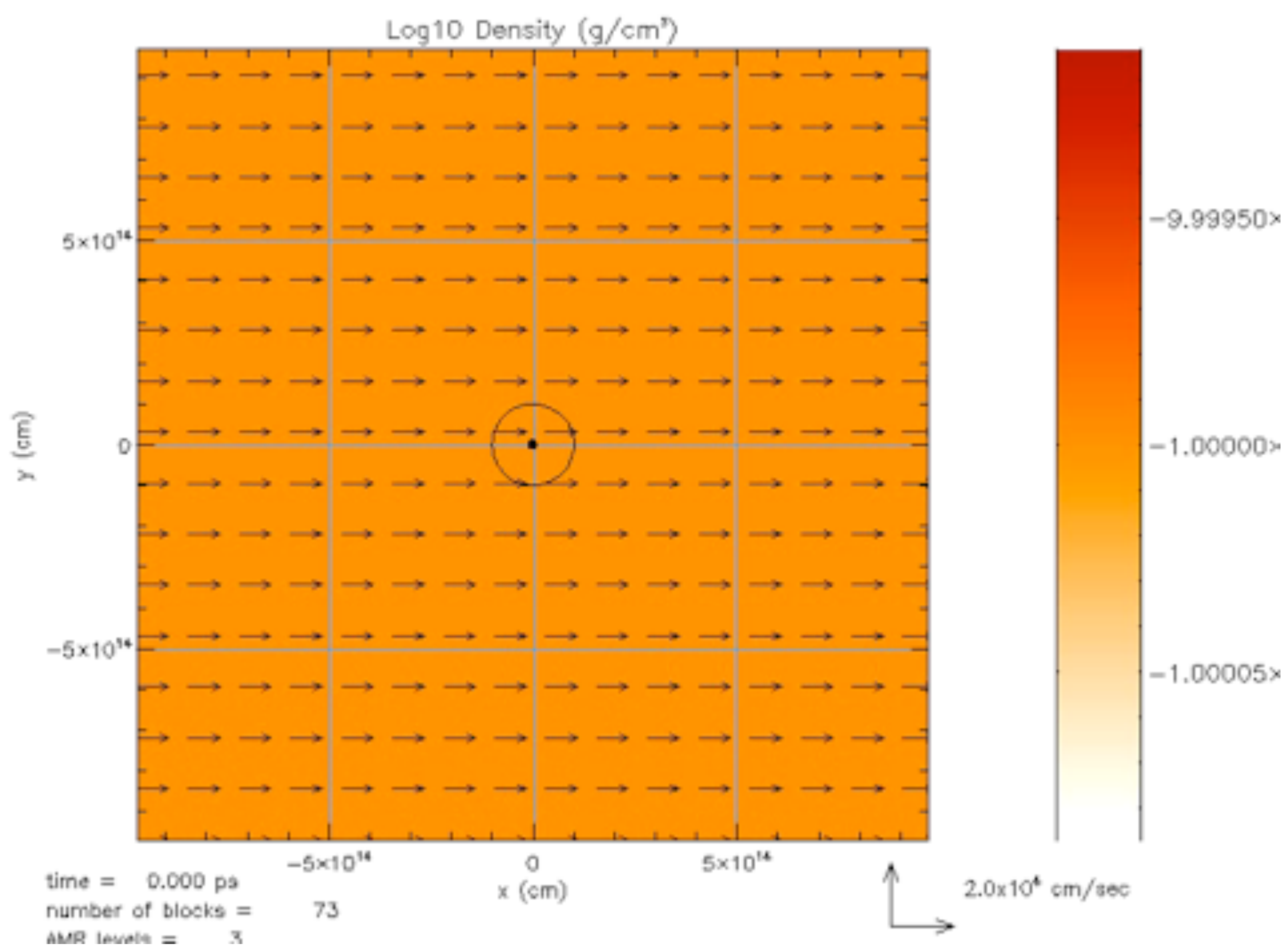


FLASH code: Sinks

Examples: gas drag on particles

- “easy” implementations of gas-particle interaction
- e.g. particle-gas drag via elastic scattering

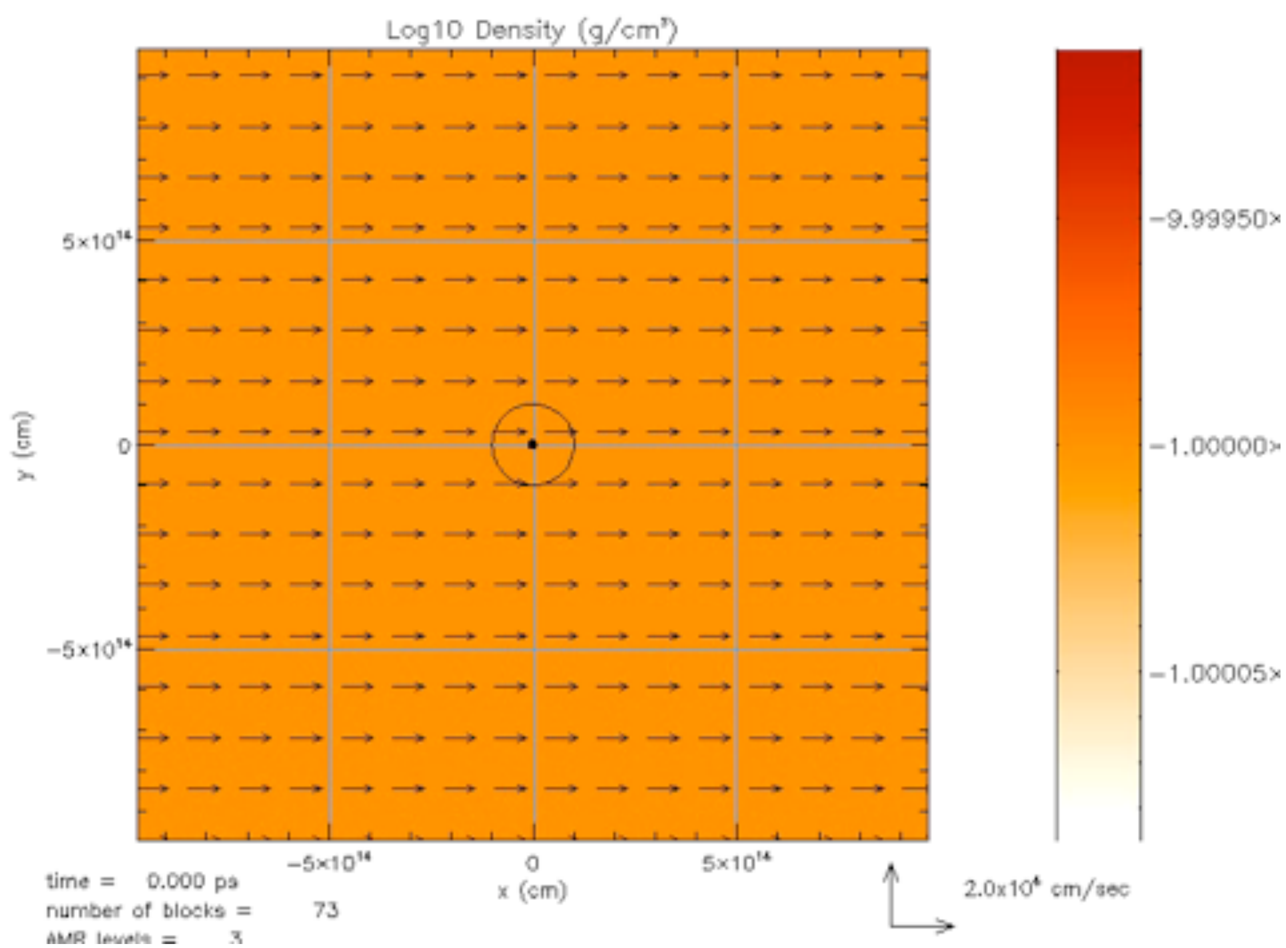




FLASH code: Sinks

Examples: gas drag on particles

- “easy” implementations of gas-particle interaction
- e.g. particle-gas drag via elastic scattering



FLASH code: Sinks

Implementation in FLASH

- see add-on manual: `docs/sinks/flash4_ug.pdf`
  (full FLASH manual: `/pfs/banerjee/flash4_ug.pdf`)

- `REQUIRES Particles/ParticlesMain/active/Sink`

- main file: `Particles_sinkCreateAccrete.F90`

  $\Longrightarrow$ modifications (e.g. SN-feedback, drag forces) should start here

FLASH code: Sinks

runtime parameters

UseSinkParticles	BOOLEAN	".false."	switch sinks on/off
sink_density_thresh	REAL	1.0e-14	density threshold for sink creation and accretion
sink_accretion_radius	REAL	1.0e14	creation and accretion radius
sink_softening_radius	REAL	1.0e14	gravitational softening radius
sink_softening_type_gas	STRING	"linear"	sink–gas softening type (options: "linear", "spline")
sink_softening_type_sinks	STRING	"spline"	sink–sink softening type (options: "linear", "spline")
sink_integrator	STRING	"leapfrog"	sink particle time integrator (options: "euler", "leapfrog", "leapfrog_cosmo")
sink_dt_factor	REAL	0.5	time step safety factor (≤ 0.5)
sink_subdt_factor	REAL	0.01	time step safety factor for sink–sink subcycling (≤ 0.5)
sink_convergingFlowCheck	BOOLEAN	.true.	creation check for converging gas flow
sink_potentialMinCheck	BOOLEAN	.true.	creation check for gravitational potential minimum
sink_jeansCheck	BOOLEAN	.true.	creation check for Jeans instability
sink_negativeEtotCheck	BOOLEAN	.true.	creation check for gravitationally bound gas
sink_GasAccretionChecks	BOOLEAN	.true.	check for bound and converging state before gas accretion
sink_merging	BOOLEAN	.false.	switch for sink particle merging
pt_maxSinksPerProc	INTEGER	100	number of sinks per processor
refineOnSinkParticles	BOOLEAN	.true.	sinks must be on highest AMR level
refineOnJeansLength	BOOLEAN	.true.	switch for refinement on Jeans length
jeans_ncells_ref	REAL	32.0	number of cells for Jeans length refinement
jeans_ncells_deref	REAL	64.0	number of cells for Jeans length de-refinement

FLASH code: Sinks

tasks

- `unitTest/SinkMomTest`
  - momentum conservation test:
    $\Longrightarrow$ collapse of a spherical cloud
    + moving particle in y-direction

- `SinkRotatingCloudCore`
  - Boss-Bodenheimer test:
    $\Longrightarrow$ collapse of a rotating cloud core with
    m = 2 density perturbation
    $\Longrightarrow$ should result 3 sink particles

- `SinkWindTest`
  $\Longrightarrow$ collapse of a cloud core with 'wind' feedback