Cluster formation simulations with AMUSE

netherlands eScience center

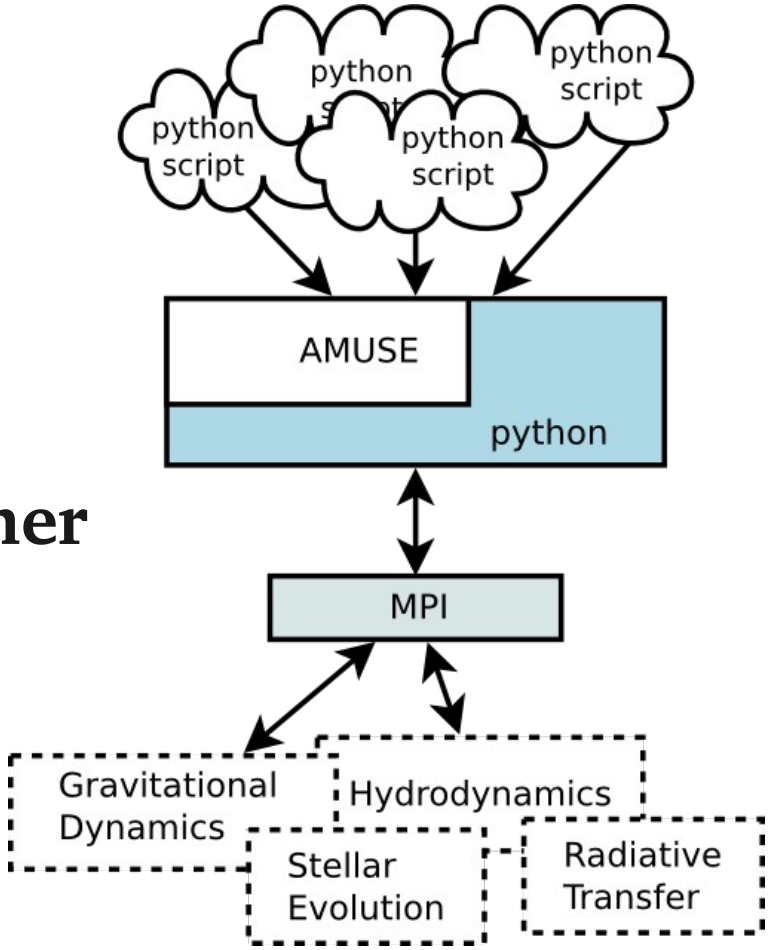
Inti Pelupessy - Arjen van Elteren – Steve McMillan

Nathan de Vries - Simon Portegies Zwart – Niels Drost

MODEST 13, Almaty workshop and summer school, 19-23 August 2013

AMUSE philosophy: "*one code is no code*"

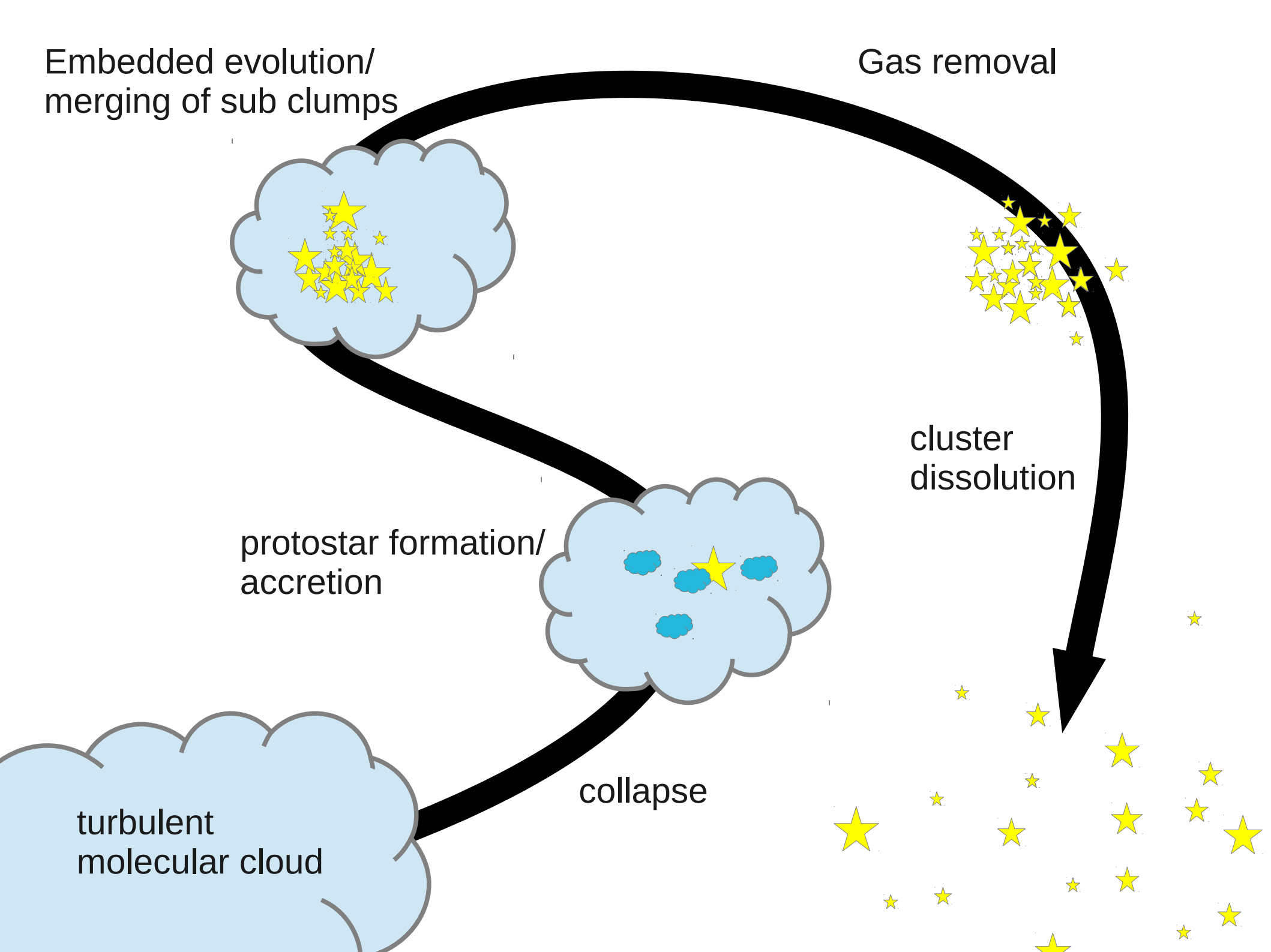
- base interface on physics
- hide code particulars
- codes should understand each other
- test everything
- run anywhere

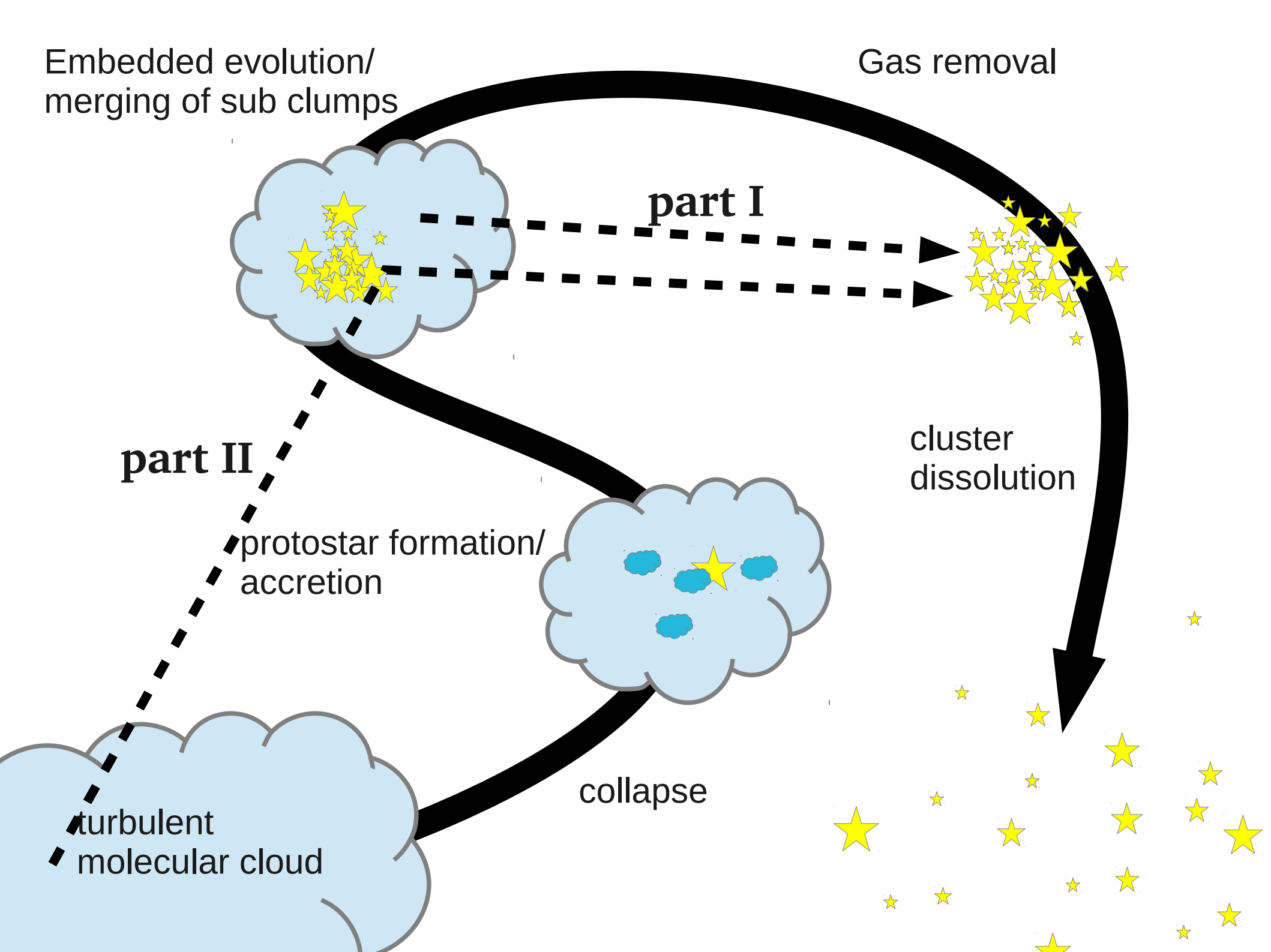


code papers: Pelupessy et al. 2013, Portegies Zwart et al. 2012

	code	language	short description	main reference
gravity	Hermite0	C++	Hermite N-body	Hut et al. (1995)
	PhiGRAPE	Fortran, MPI/GPU	Hermite N-body	Harfst et al. (2007)
	ph4	C++, MPI/GPU	Hermite N-body	McMillan, in prep.
	BHTree	C++	Barnes-Hut treecode	Barnes & Hut (1986)
	Octgrav	C++, CUDA	Barnes-Hut treecode	Gaburov et al. (2010)
	Bonsai	C++, CUDA	Barnes-Hut treecode	Bédorf et al. (2012)
	Twobody	Python	Kepler solver	Bate et al. (1971)
	Huayno	C, OpenMP/OpenCL	Hamiltonian splitting	Pelupessy et al. (2012)
	SmallN	C++	regularized solver	Portegies Zwart et al. (1999)
	Mercury	Fortran	symplectic planetary integrator	Chambers (1999)
	Mikkola	Fortran	relativistic regularization	Mikkola & Merritt (2008)
	MI6	C++, MPI/GPU	Hermite with Post-Newtonian terms	Iwasawa et al. (2011)
	Pikachu	C++, CUDA	Hybrid Barnes-Hut/Hermite	Iwasawa et al., in prep.
	Brutus	C++, MPI	arbitrary precision Bulirsch-Stoer	Boekholt & Portegies Zwart, in prep.
	HiGPUs	C++, CUDA	Hermite N-body	Capuzzo-Dolcetta et al. (2013)
	Tupan	Python, OpenCL	Symplectic N-body, Post-Newtonian	Ferrari, in prep.
	MMC	Fortran	Monte-Carlo gravitational dynamics	Giersz (2006)
stellar evolution	SSE	Fortran	stellar evolution fits	Hurley et al. (2000)
	Evtwin	Fortran	Henyey stellar evolution	Glebbeek et al. (2008)
	MESA	Fortran, OpenMP	Henyey stellar evolution	Paxton et al. (2011)
	BSE	Fortran	binary evolution	Hurley et al. (2000)
	SeBa	C++	stellar and binary evolution	Portegies Zwart et al. (2001)
hydro-dynamics	Fi	Fortran, OpenMP	TreeSPH	Pelupessy (2005)
	Gadget-2	C, MPI	TreeSPH	Springel (2005)
	Capreole	Fortran, MPI	Finite volume grid hydrodynamics	Mellema et al. (1991)
	Athena3D	C, MPI	Finite volume grid hydrodynamics	Stone et al. (2008)
	MPIAMRVAC	Fortran, MPI	AMR code for conservation laws	Keppens et al. (2012)
radiative	Simplex	C++, MPI	Rad. transport on Delaunay grid	Paardekooper et al. (2010)
	SPHRAY	Fortran	Monte Carlo on SPH particles	Altay et al. (2008)
	Mocassin	Fortran	Monte Carlo, steady state	Ercolano et al. (2003)
misc.	MMAMS	C++	stellar mergers by entropy sorting	Gaburov et al. (2008)
	Hop	C++	particle group finder	Eisenstein & Hut (1998)
	FractalCluster	Fortran	Fractal cluster generator	Goodwin & Whitworth (2004)
	Halogen	C	Halo distribution functions	Zemp et al. (2008)
	GalactICS	C	Galaxy model generator	Widrow et al. (2008)

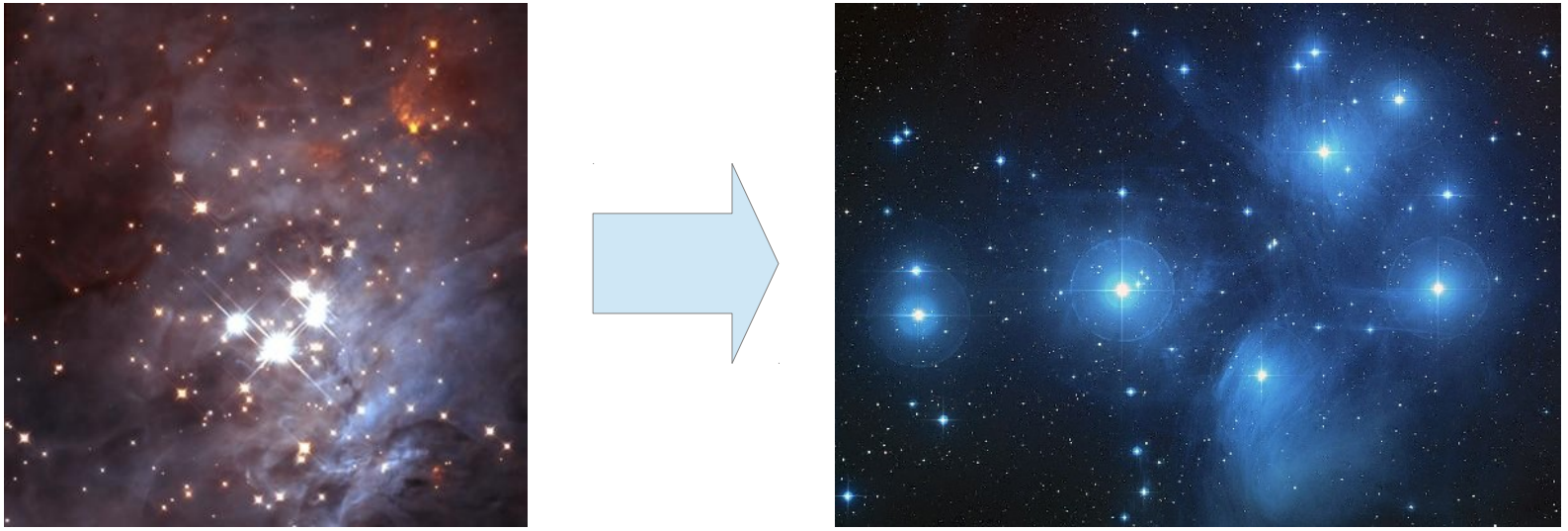
Table 1. Overview of community codes included in the public release of AMUSE. Table lists the name of the code, the language it is written in and the parallelization model, a short description and a reference to either the code paper or a description of the method.


Embedded evolution/
merging of sub clumps
Gas removal
cluster
dissolution
protostar formation/
accretion
collapse
turbulent
molecular cloud

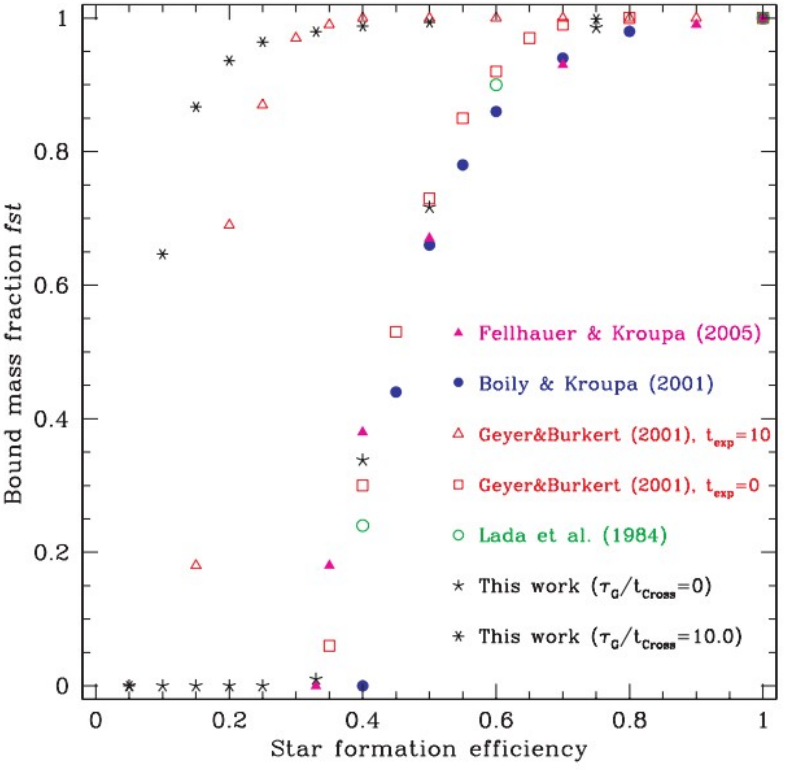

Embedded evolution/
merging of sub clumps
Gas removal
part I
part II
protostar formation/
accretion
cluster
dissolution
collapse
turbulent
molecular cloud

(I) embedded cluster evolution

Application: Early Cluster Dissolution

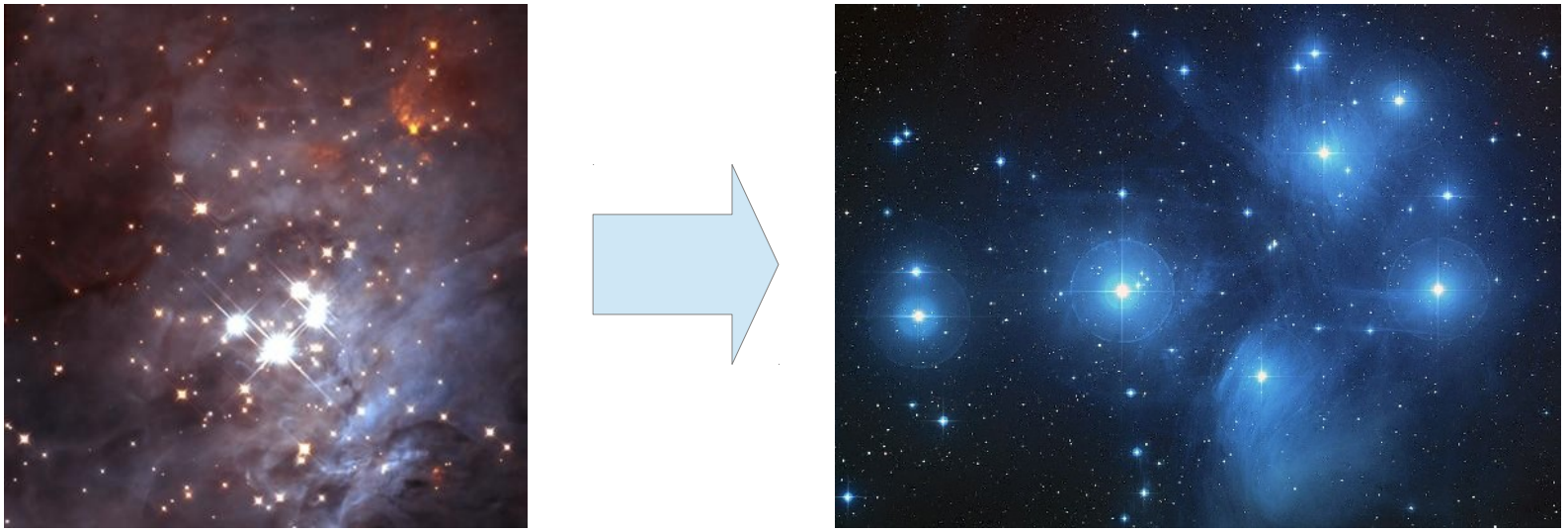


Young clusters blow away remnant gas affecting cluster structure,potentially unbinding the cluster:



Baumgardt & Kroupa 2007

Application: Early Cluster Dissolution

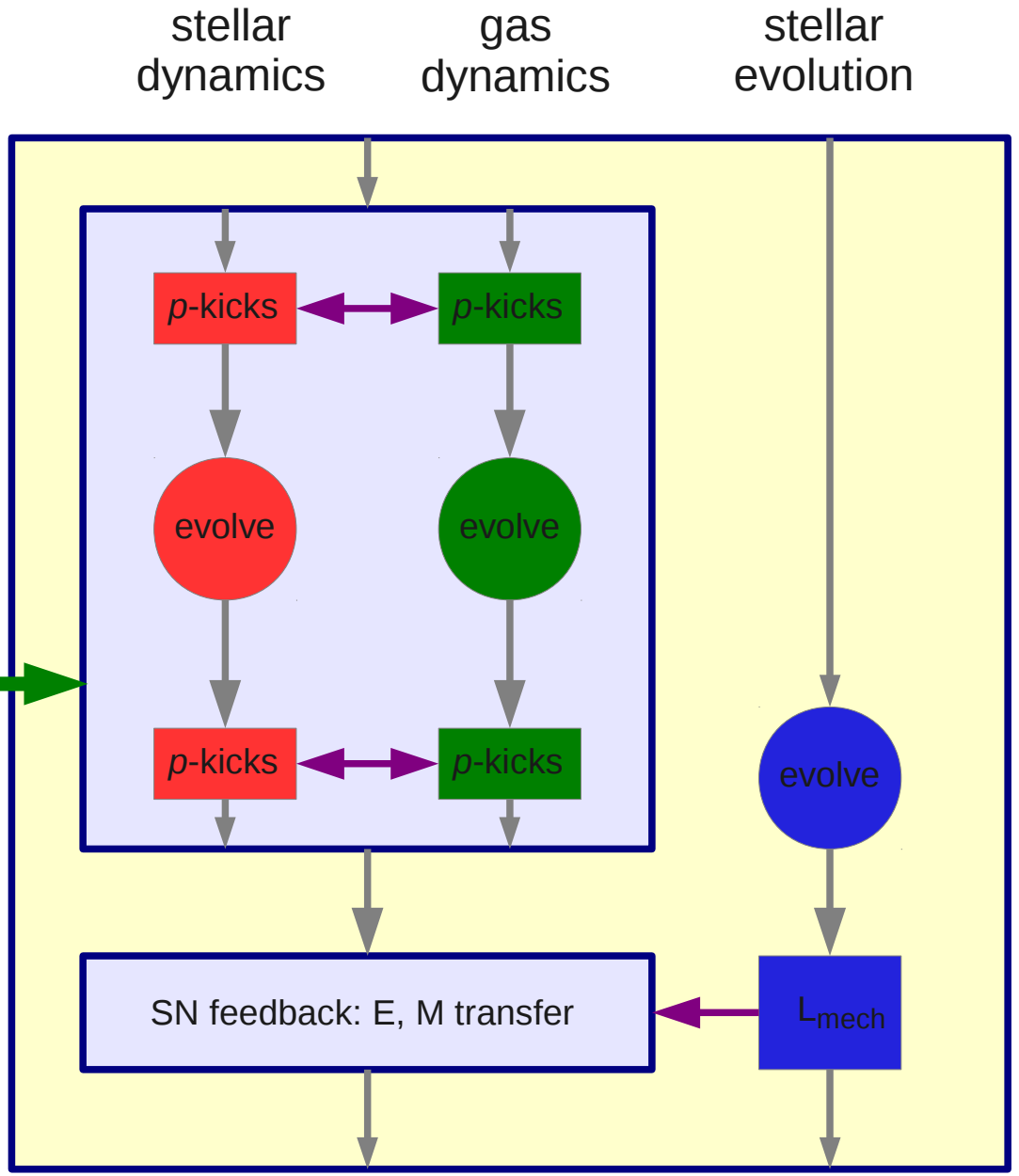


- collisional stellar dynamics
- gas dynamics
- stellar evolution
- stellar wind and supernova feedback

} AMUSE grav_gas_sse

(Pelupessy 2012, MNRAS, 420, 1503)

grav_gas_sse integrator:



"Bridge"

Red: stellar dynamics

green: gas dynamics

blue: stellar evolution

purple arrows: interactions

circle: legacy code

rectangle: python

grav_gas_sse integrator:

actual codes used:

- Stellar dynamics: phiGRAPE (Nbody, GPU)
- Hydrodynamics: Gadget 2 (SPH, MPI parallel)
- Stellar evolution: SSE
- Hydro self gravity, gas ↔ star interactions:
  Octgrav (GPU treecode)

Other parts of the script (e.g. recipes for mechanical luminosity, injection of particles) and the 'glue' are implemented in AMUSE (python)

Example run:

- 1000 star embedded cluster

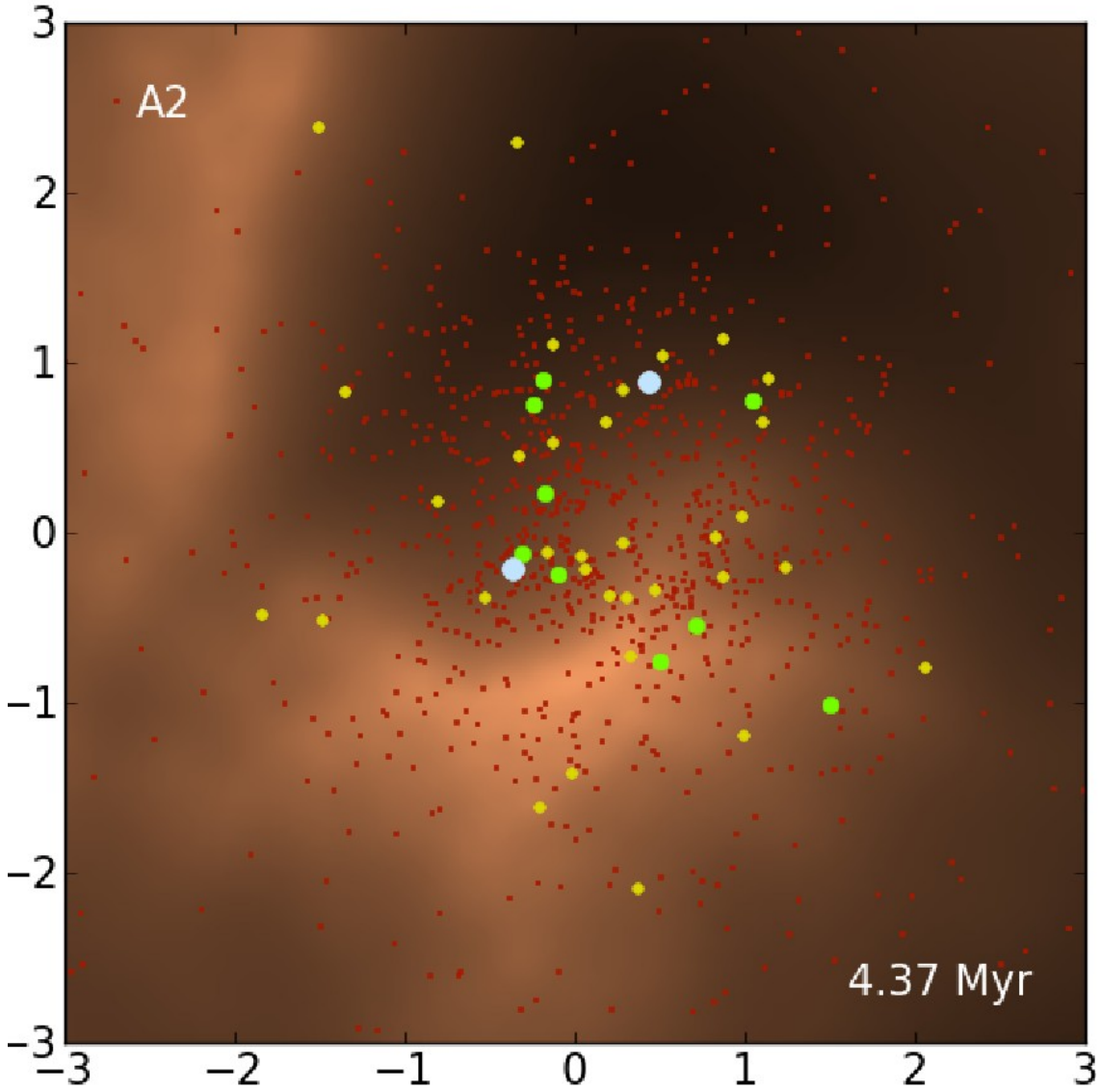
- gas densities ~ $10^{4\text{-}5}$ cm^{-3}

- Plummer sphere

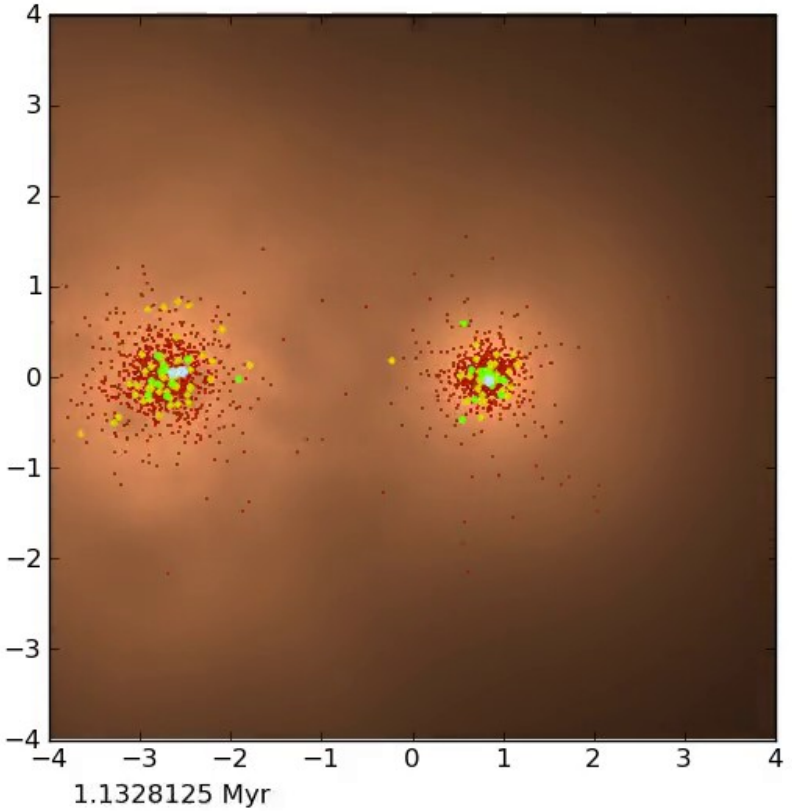
- constant SFE

- Salpeter IMF (0.1-100 M_{sun})

- 30 Myr


A2
4.37 Myr
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

Two (almost) equal clusters on collision course:



SFE = 0.3, 2x1000 stars, most massive star: 37 M_{sun} (left) and 19 M_{sun}(right)

Conclusions

- gas dynamics is an important driver of early cluster evolution: determining factors are the relative *distribution* of gas and stars, the *type* and *timing* of feedback and the *kinematic state* of the stellar component,

- mass segregation can be a relic from the embedded evolution,

- gas clearing affects the stellar mass function of the remnant cluster,

- gas loss will also be crucial for tidal disruption scenarios or hierarchical star formation

(II) self consistent star formation simulations (WIP)

Recipe for a cluster formation simulation

- turbulent cloud subject to gravitational collapse
- density criterion for formation of star forming cores [stopping condition]

- sink particles accrete from high density gas [sink particles]
- sink particles generate stars once: $t > t_{\rm pms}$

- accreting stellar objects passed to stellar evolution code & passed on to collisional gravity code [bridge]

- stellar evolution determines L_{mech}

- feedback blows away gas [grav_gas_sse]

blue = corresponding AMUSE concept

How to model the thermodynamic state of the gas?

→ simple adiabatic (as in the embedded simulations) doesn't allow for collapse

→ simplified Equation of State: isothermal (Bonnell 2003) or barotropic EOS have been shown to be good approximations for GMC type clouds.

However: incompatible with feedback, assume equilibrium

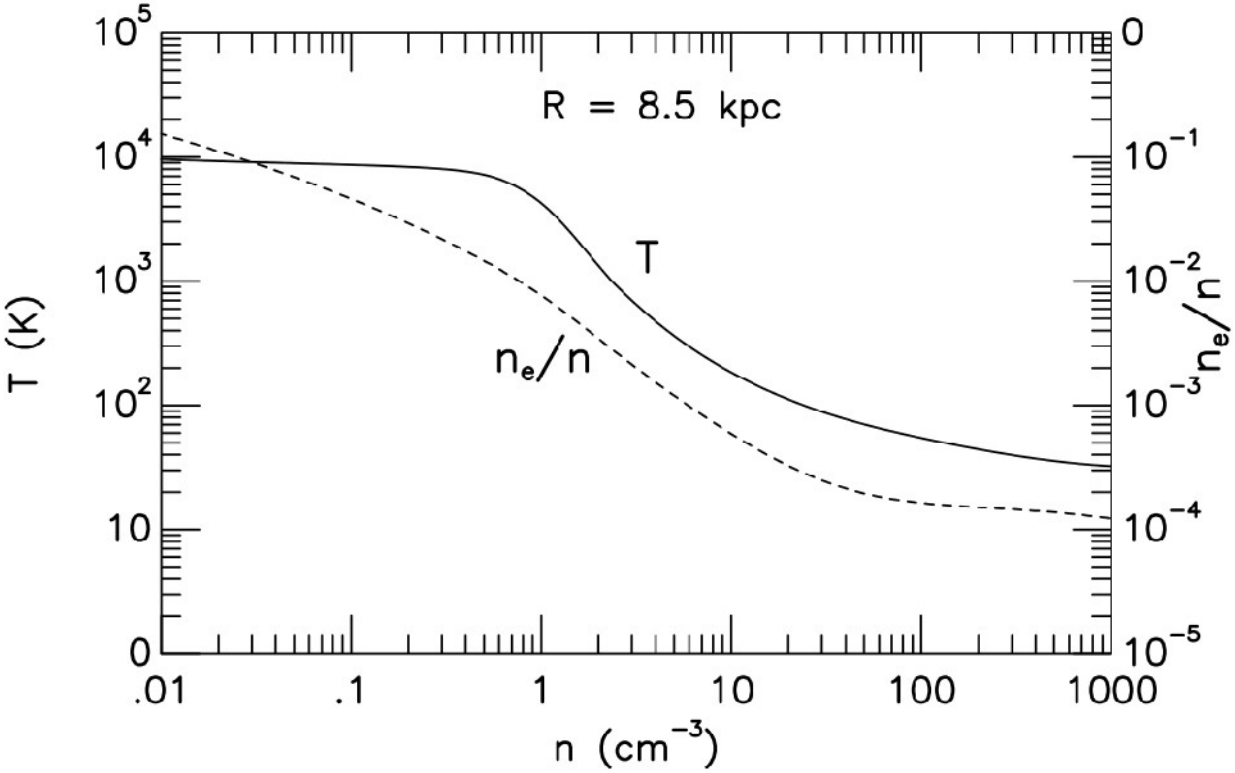
→ self consistent models exist for of chemistry and thermal balance (e.g. Wolfire 2003, Kazandjian 2012, Cloudy etc).

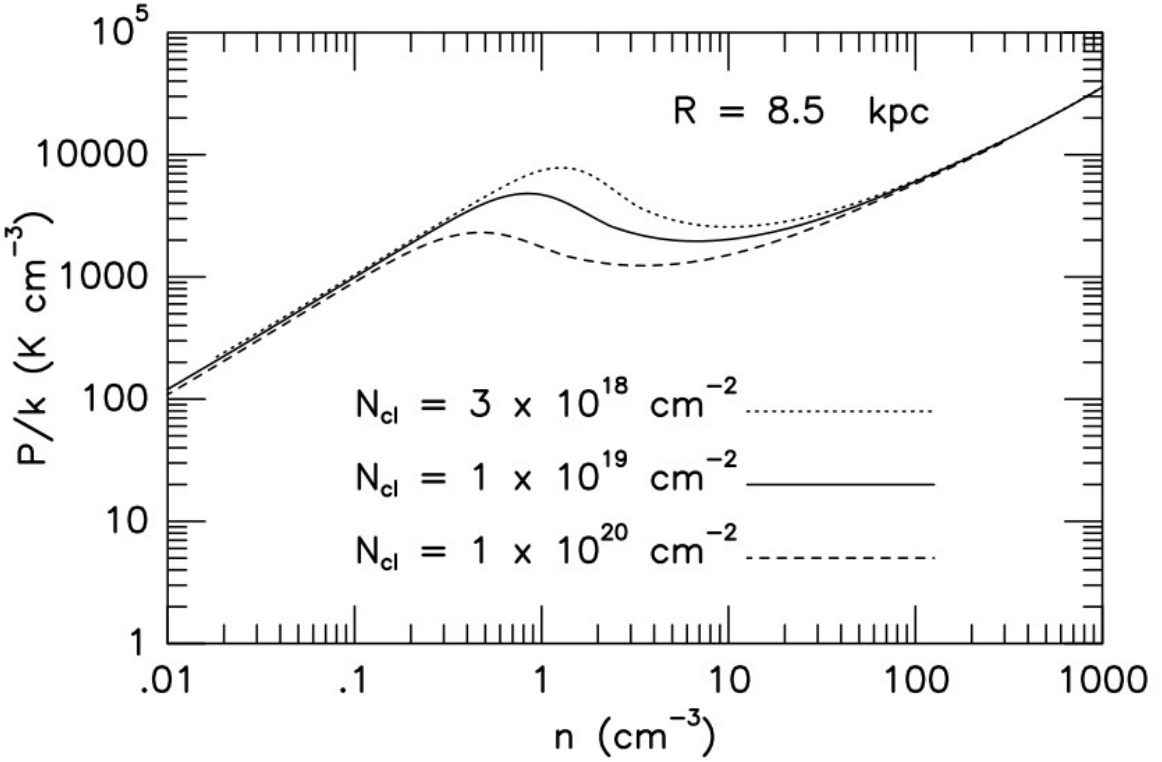
However: computationally expensive, calculate equilibria, introduces dependency on abundances, atomic rates etc

idea: formulate and test a simplified thermal model first

Wolfire 2003 model

- photoelectric/ soft X-ray heating
- atomic line cooling
- PAHs
- cosmic ray ionization
- calculates chemical and thermal equilibrium





Simple non-equilibrium model

equilibrium n-T relation:

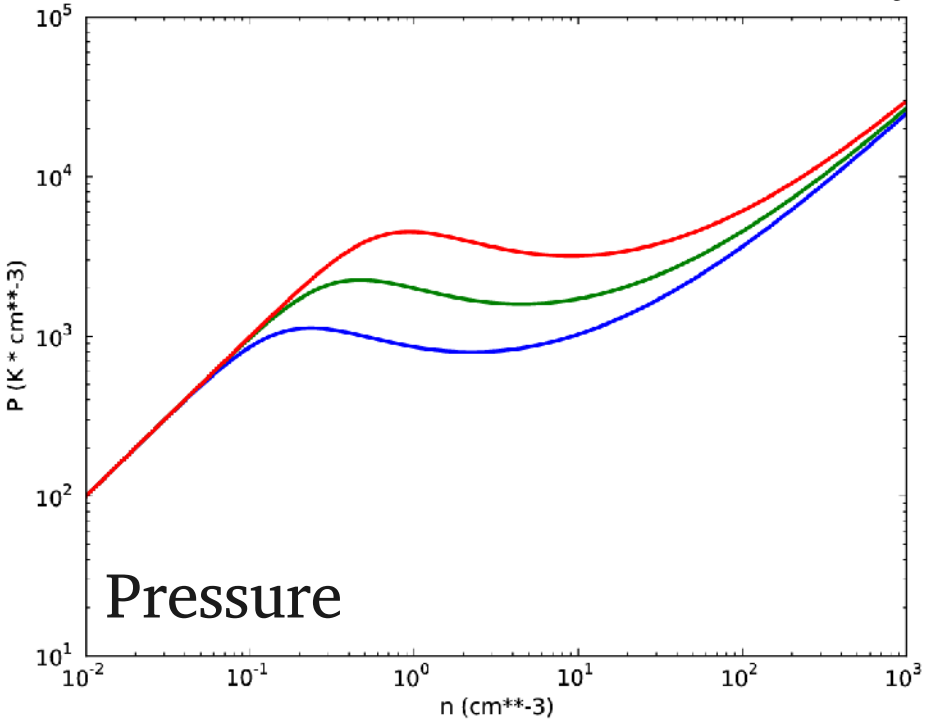
$$T_{\rm eq} = T_{\rm min} + \frac{T_0 - T_{\rm min}}{1 + \log(1 + \rho/\rho_0)^{\alpha}}$$

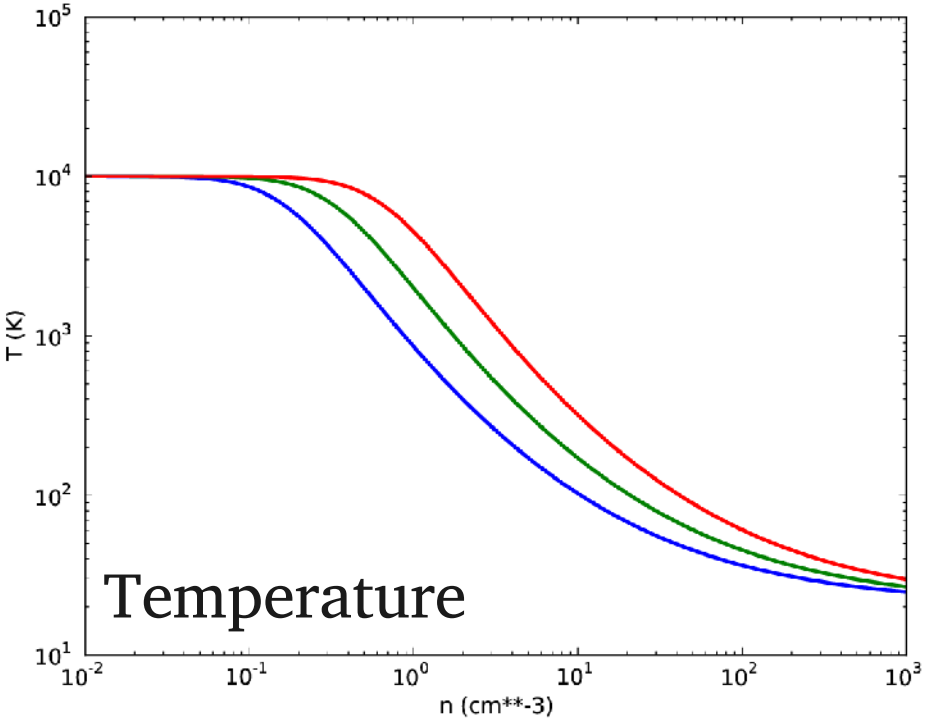
- thermal evolution equation (for internal energy u):

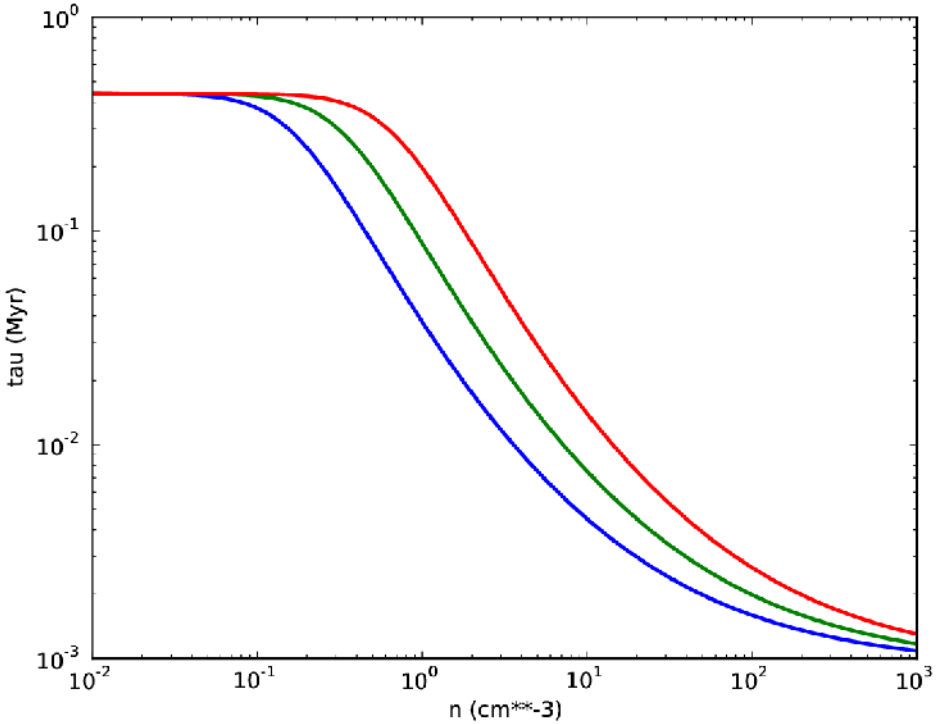
$$\frac{du}{dt} = -\tau(\rho)(u - u_{\rm eq}(\rho)) + \left(\frac{du}{dt}\right)_{\rm ad}$$

- τ estimated using FUV heating rate

Summary of the model

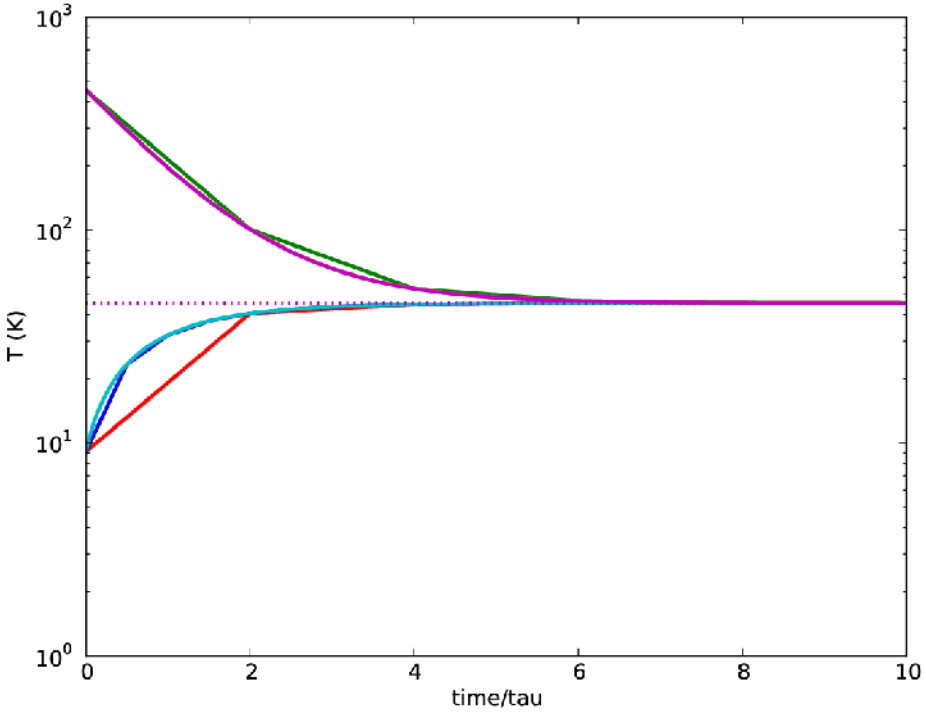


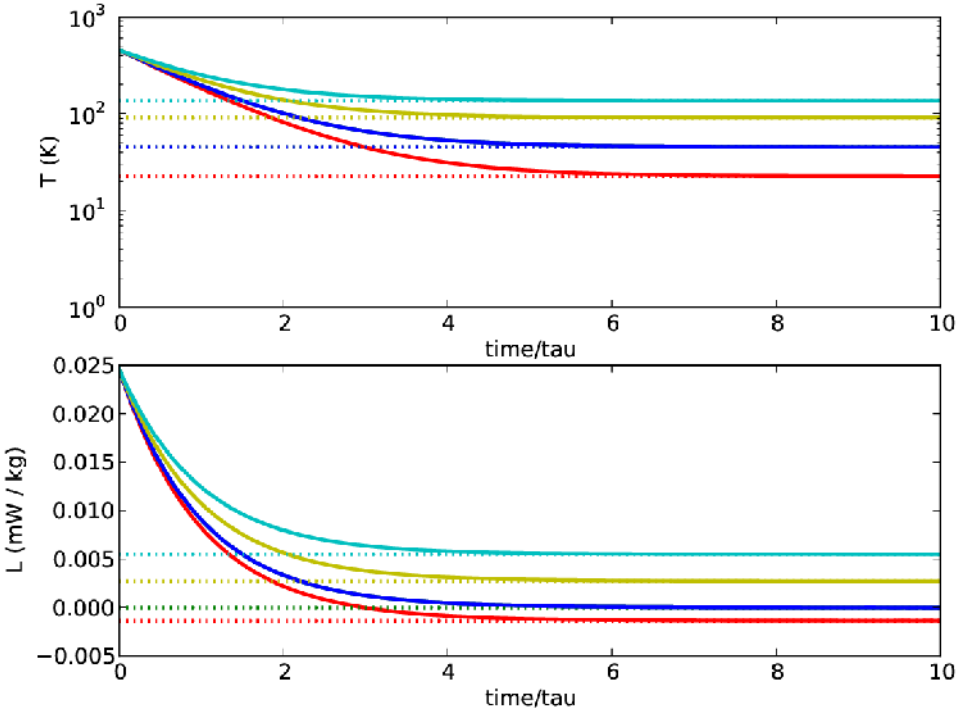




equilibrium timescale

Time evolution and L_{net}





timestepping

varying $\left(\frac{du}{dt}\right)_{\text{ad}}$

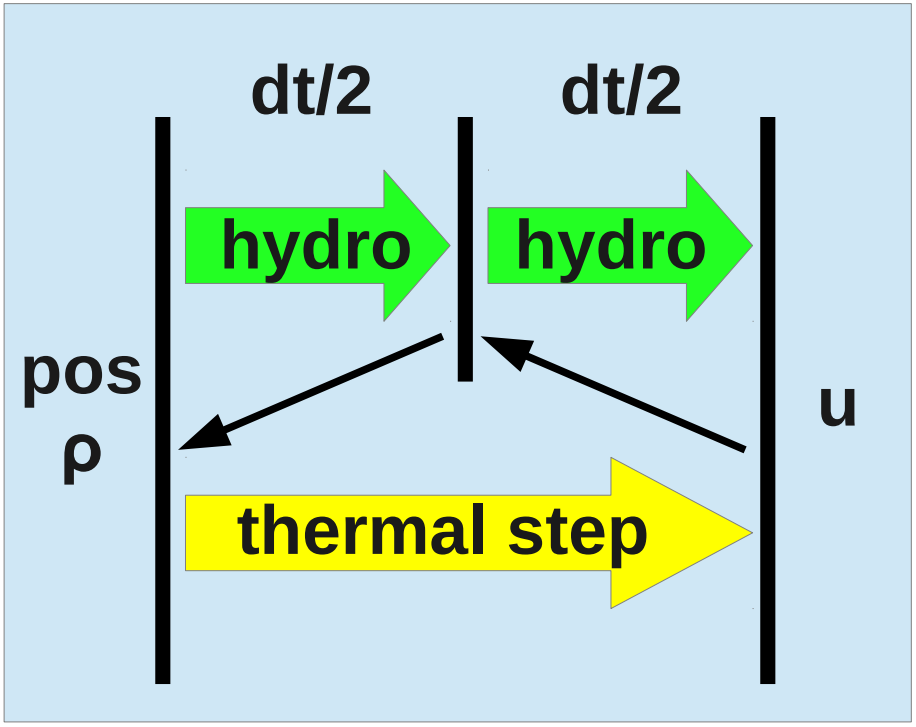
Operator splitting for hydrodynamic and thermal evolution

- operator splitting:
  isothermal or adiabatic hydro by hydrodynamic code
  thermal and radiation by e.g. radiative code

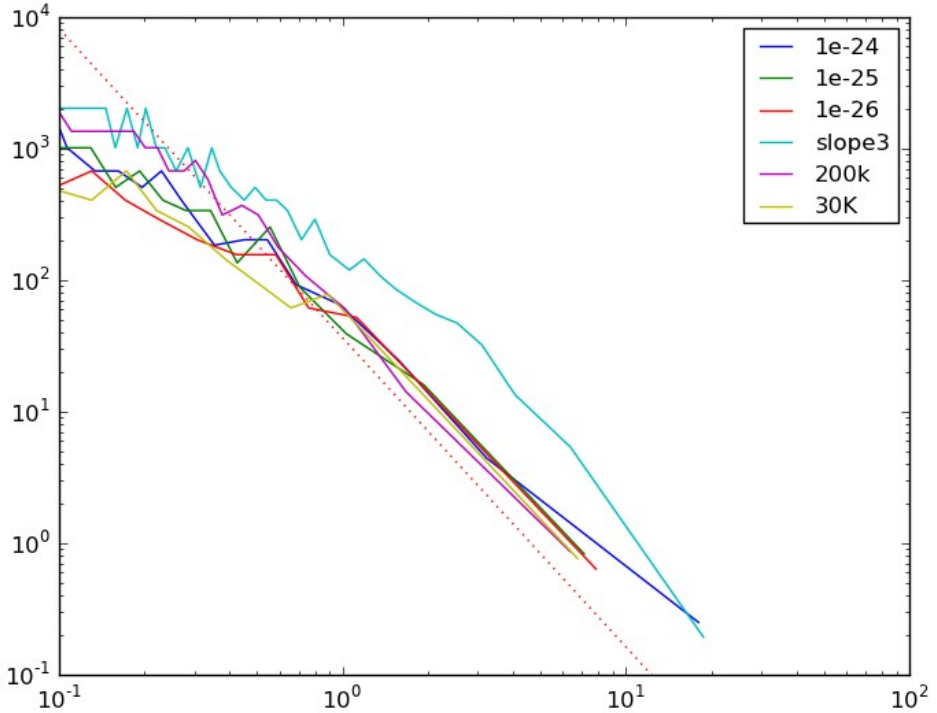
- again 'leapfrog'-type integration

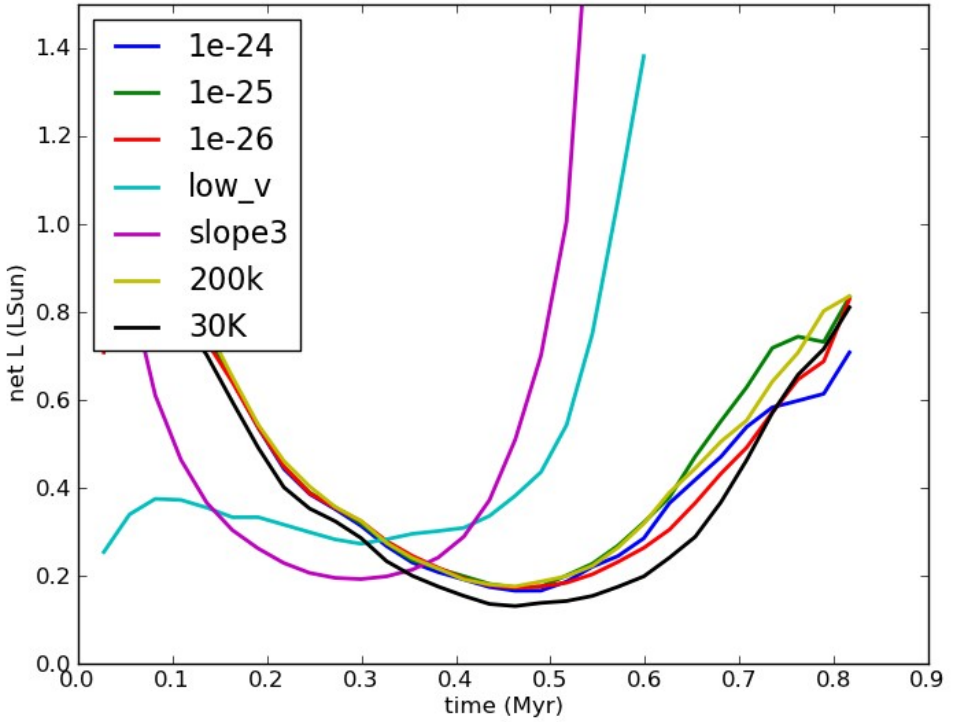
- hydro evolves:
  positions, velocities

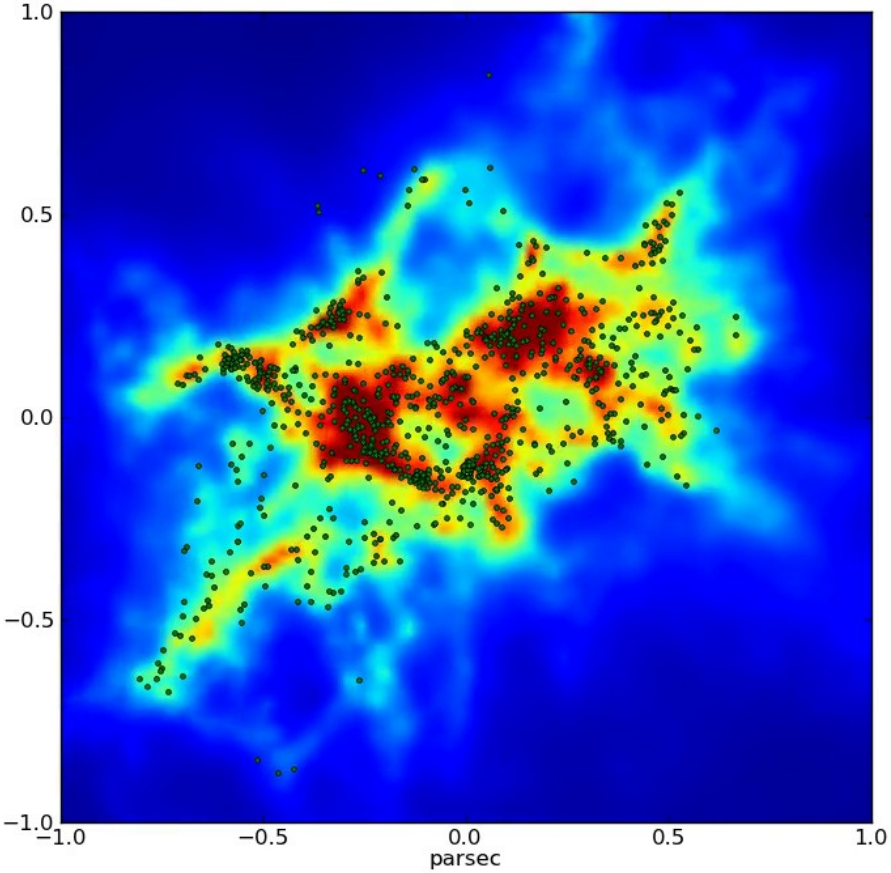
- thermal code evolves:
  internal energy, ionization



Example: collapse test







Conclusions

- the AMUSE interface allows you to use codes as building blocks for development of astrophysical simulations or solvers,
- such constructs can be ad-hoc solutions, or rigorously derived (such as BRIDGE),
- new solvers can be used as building blocks for more complicated solvers
- this allows for cross verification of different codes or between different numerical methods,
- AMUSE provides a flexible toolbox to quickly try out and develop models

Amuse install options:

- binary releases (linux 64 bit, Mac OS 32 bit, 64 bit)
- Virtual Box image (OpenSUSE 64 bit)
- source download:
  instructions compile prerequisites and the code itself are available
- svn repository

www.amusecode.org