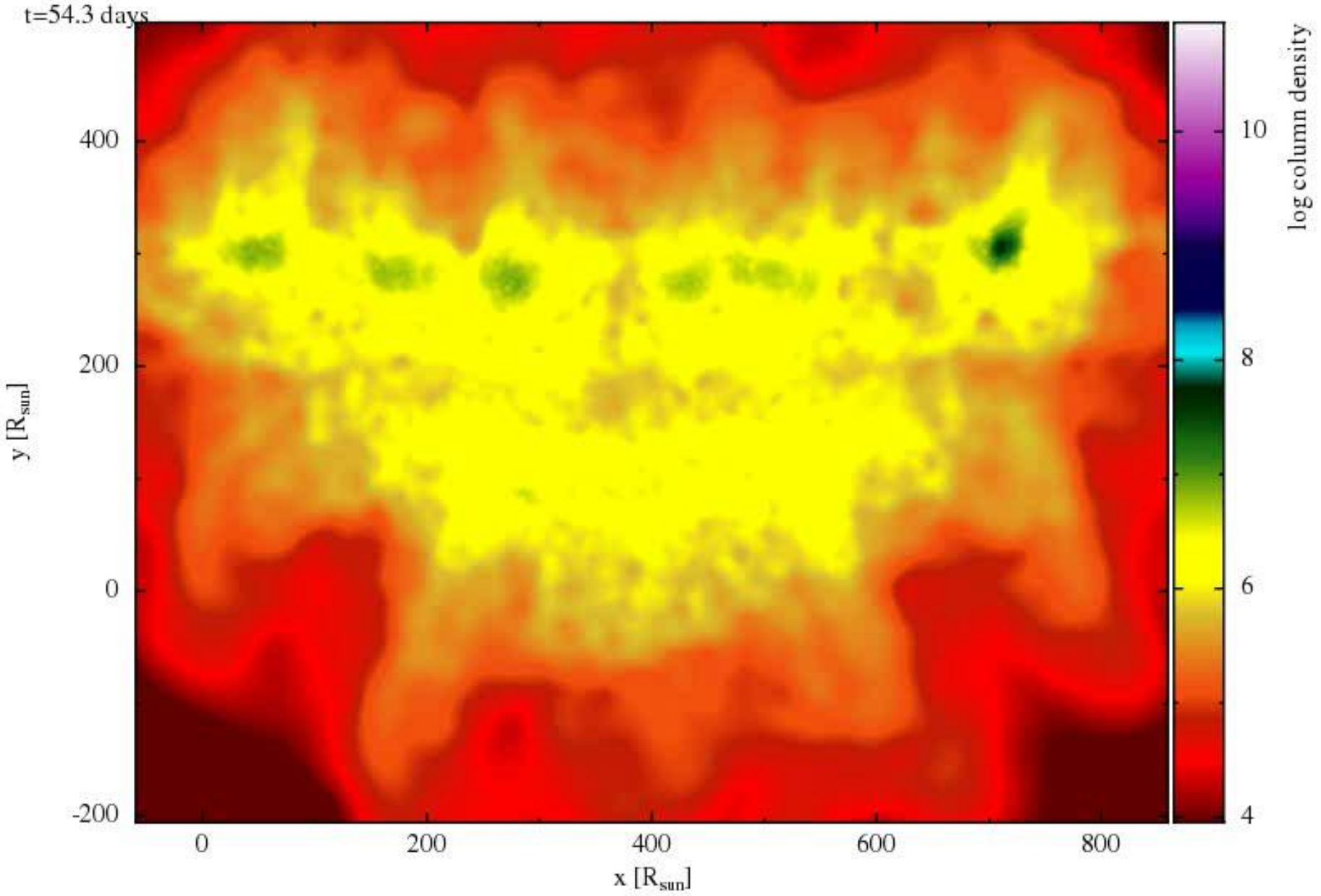


(N=13k particles , tabulated EOS, no radiative cooling, 14 seconds wallclock time)

SPH Basics

- SPH = Smoothed Particle Hydrodynamics
- Lagrangian formuation: particles trace the fluid flow
- Each SPH particle is tagged with
  - Physical quantities
    - Mass m
    - Cartesian coordinates x, y, z
    - Velocity components v_x, v_y, v_z
    - Specific internal energy u or an entropic variable A
    - Composition
  - Numerical quantity: "smoothing length" h gives (half) radius of the mass distribution of a particle
- From these quantities, others can be constructed w/ help from kernel W:
  - Density $\rho_i = \sum_j m_j W(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h_i)$
  - Temperature & pressure (need EOS), acceleration
- Also can include dark matter/compact object/core particles that interact gravitationally but not hydrodynamically with the rest of the system

See Rosswog (2009) for a detailed review.

As in an *N*-body code

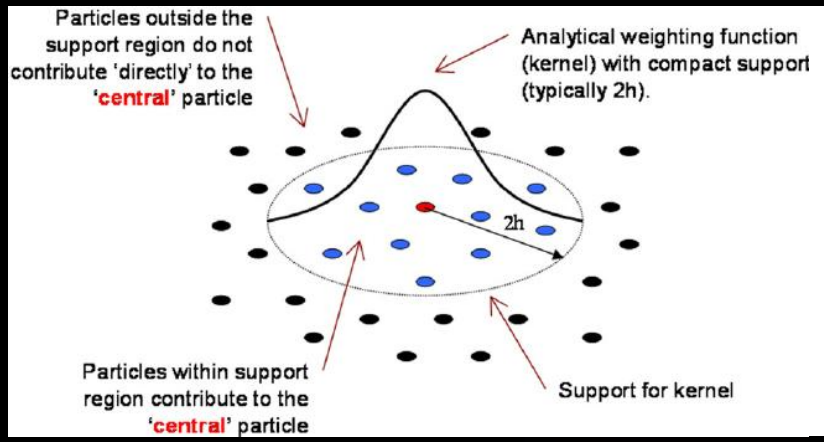


Image from Karekal et al. (2011)

Publicly available SPH codes (w/ incomplete comments on each)

- *Gadget-2* / *Gadget-3* (http://wwwmpa.mpa-garching.mpg.de/gadget/)
  - Primarily cosmological simulations
  - Volker Springel is the primary/original code developer
  - Large user base
- *PhantomSPH* (https://phantomsph.bitbucket.io/)
  - From the makers of SPLASH
  - Well documented
  - MHD ("The only time magnetic fields aren't important is when they are zero." ---Joe Monaghan)
- *StarCrash* (http://ciera.northwestern.edu/StarCrash/)
  - FFTW calculation
  - Old equations of motion (i.e. not derived from a Lagrangian)
- *Starsmasher* (http://jalombar.github.io/starsmasher/)
  - Similar I/O to StarCrash
  - GPU calculation of gravity
  - Starting to be better documented

Starsmasher

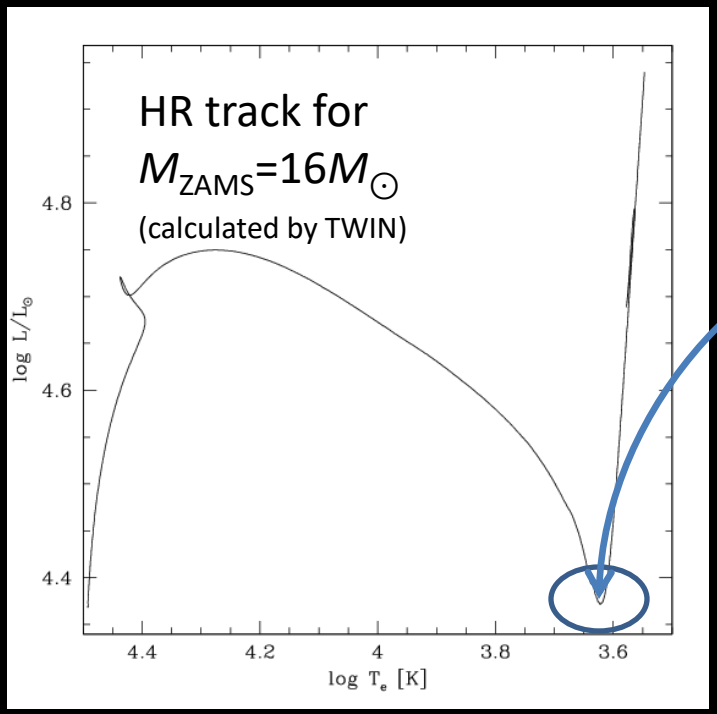
- Equations of motion derived from Lagrangian (Gaburov et al. 2010)
- Parallelized
  - on CPUs (the hydrodynamics and bookkeeping)
  - on GPUs (gravity) … thanks to Evghenii Gaburov
- Direct summation gravity ensures energy, angular momentum, and momentum conservation
- Options
  - Tabulated equation of state
  - Approximate radiative cooling (important when generating light curves)
  - Three different kernels to choose from

The cluster that we run most simulations on has 6 nodes and 4 GPUs per node; usually one job per node

Lots of undergraduates involved, e.g. Travis Court ‘18

How to model a (synchronized) binary
Step 0: Pick your stars

- Decide on realistically modeled stars or polytropes. E.g.,

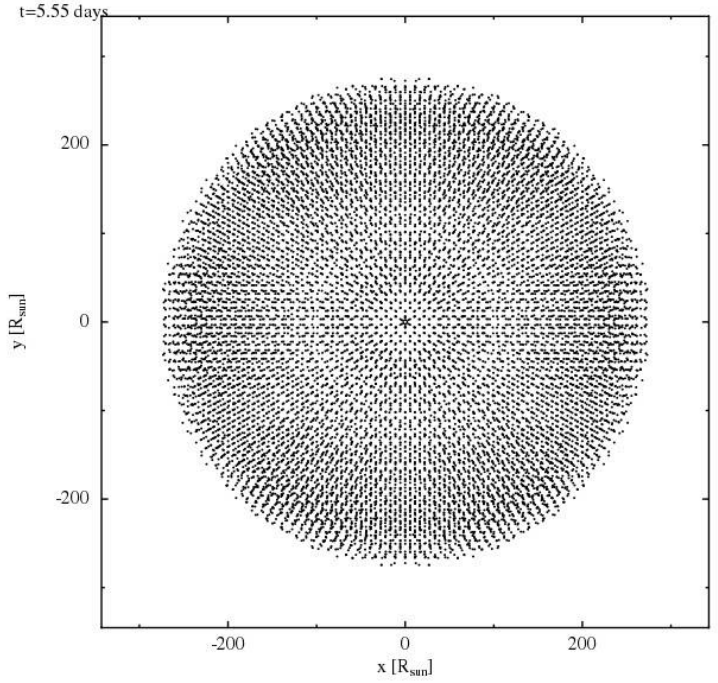


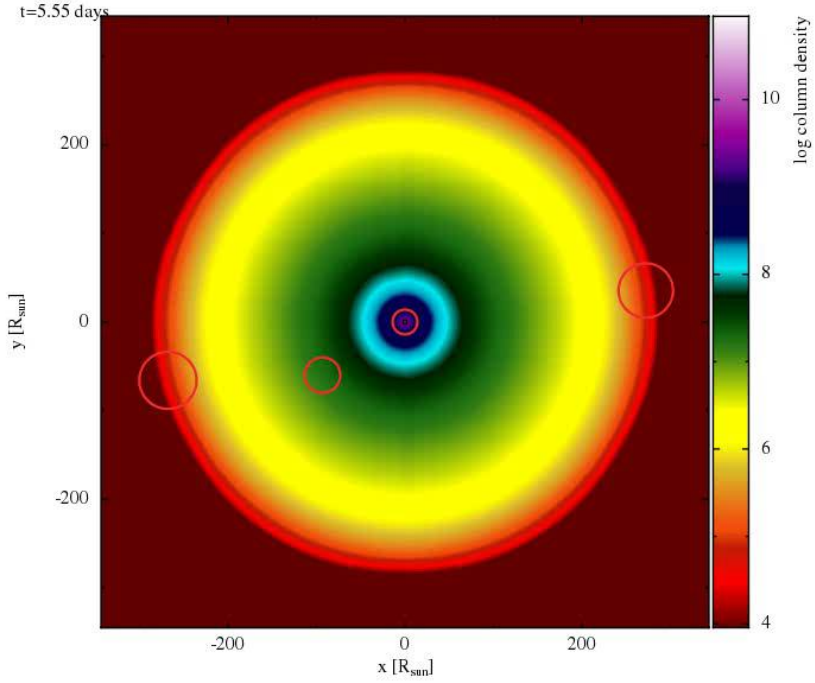
Primary:
Age = 10.006 Myr
M = 15.7 $M_{\odot}$
R = 292 $R_{\odot}$

Secondary:
Pre-MS modelled with n=1.5 density and pressure profiles but tabulated EOS
M = 1 $M_{\odot}$

How to model a (synchronized) binary
Step 1: Relax single stars

- Generate SPH models of polytropes or realistically modeled stars.
- Our example primary is given a ~5$M_\odot$ core particle (representing He core)





80k particles

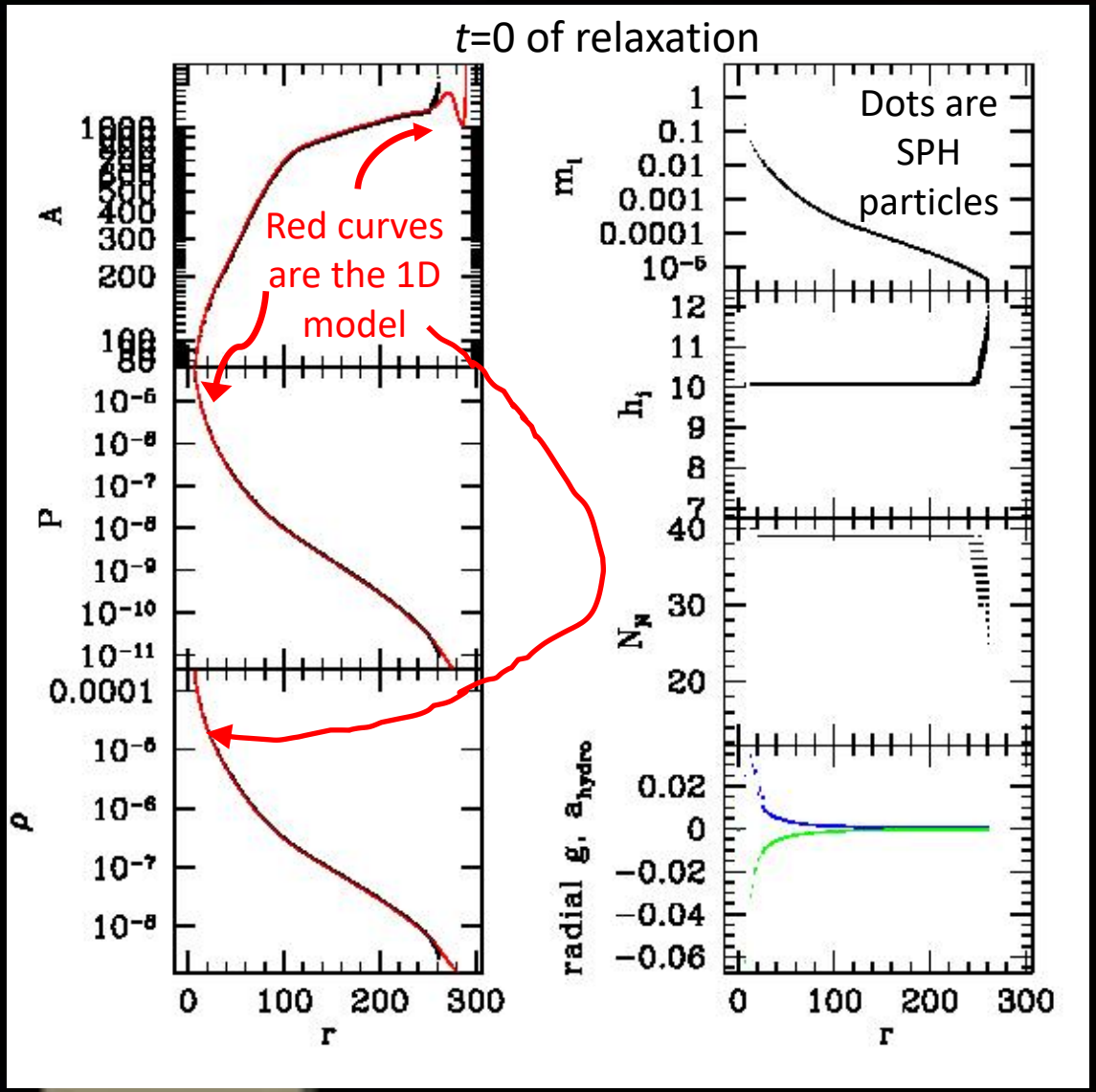
Red circles in movie on right show kernel sizes

$P/\rho^{(5/3)}$ (even though EOS is not monatomic ideal gas)

Pressure P

Density ρ

Units:
$G=M_\odot=R_\odot=1$



Particle mass m_i

Smoothing length h_i

Neighbor number N_N

a_{Hydro} (blue) & gravitational acceleration g (green)

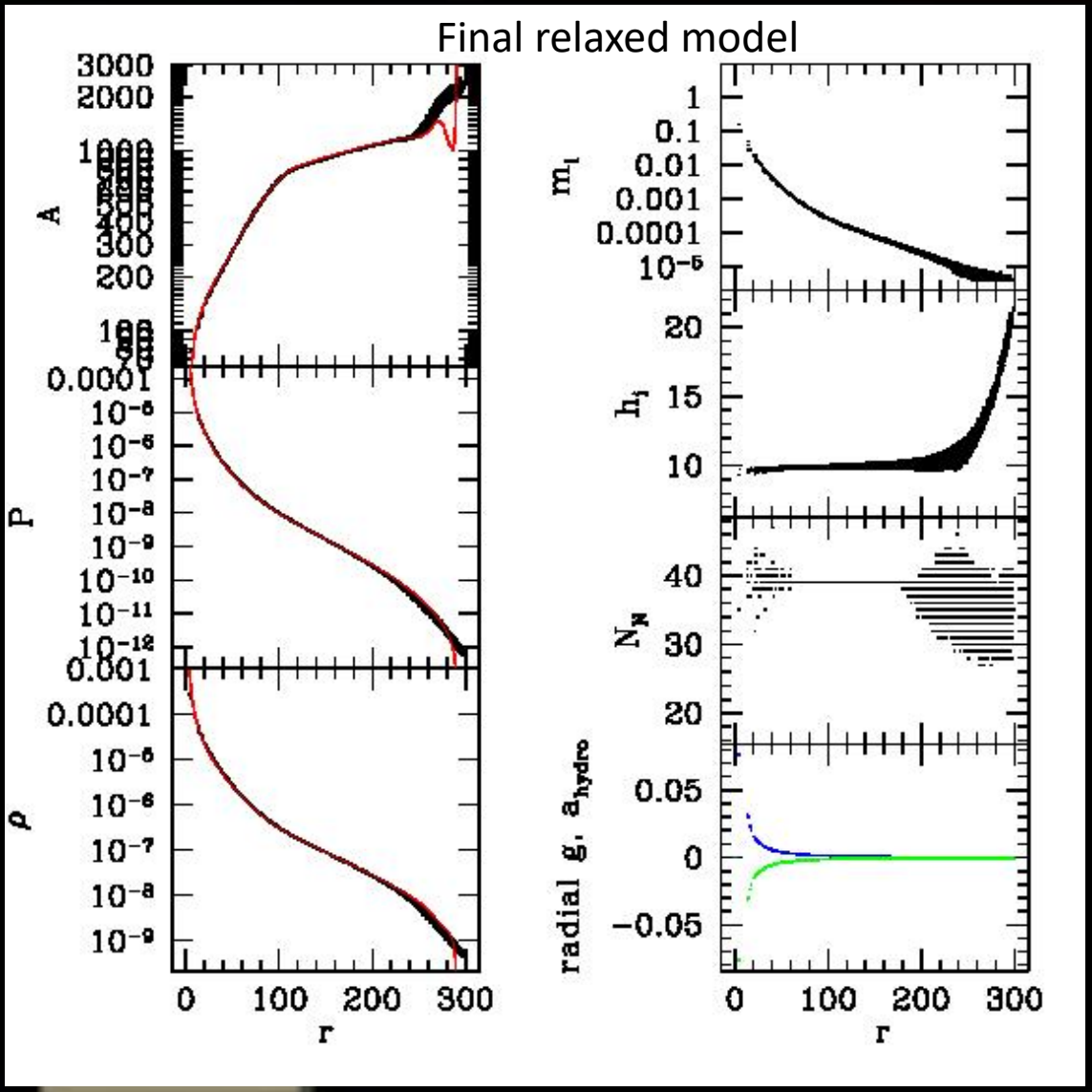
$P/\rho^{(5/3)}$ (even though EOS is not monatomic ideal gas)

Pressure P

Density ρ

Units:
$G=M_\odot=R_\odot=1$

Final relaxed model



Particle mass m_i

Smoothing length h_i

Neighbor number N_N

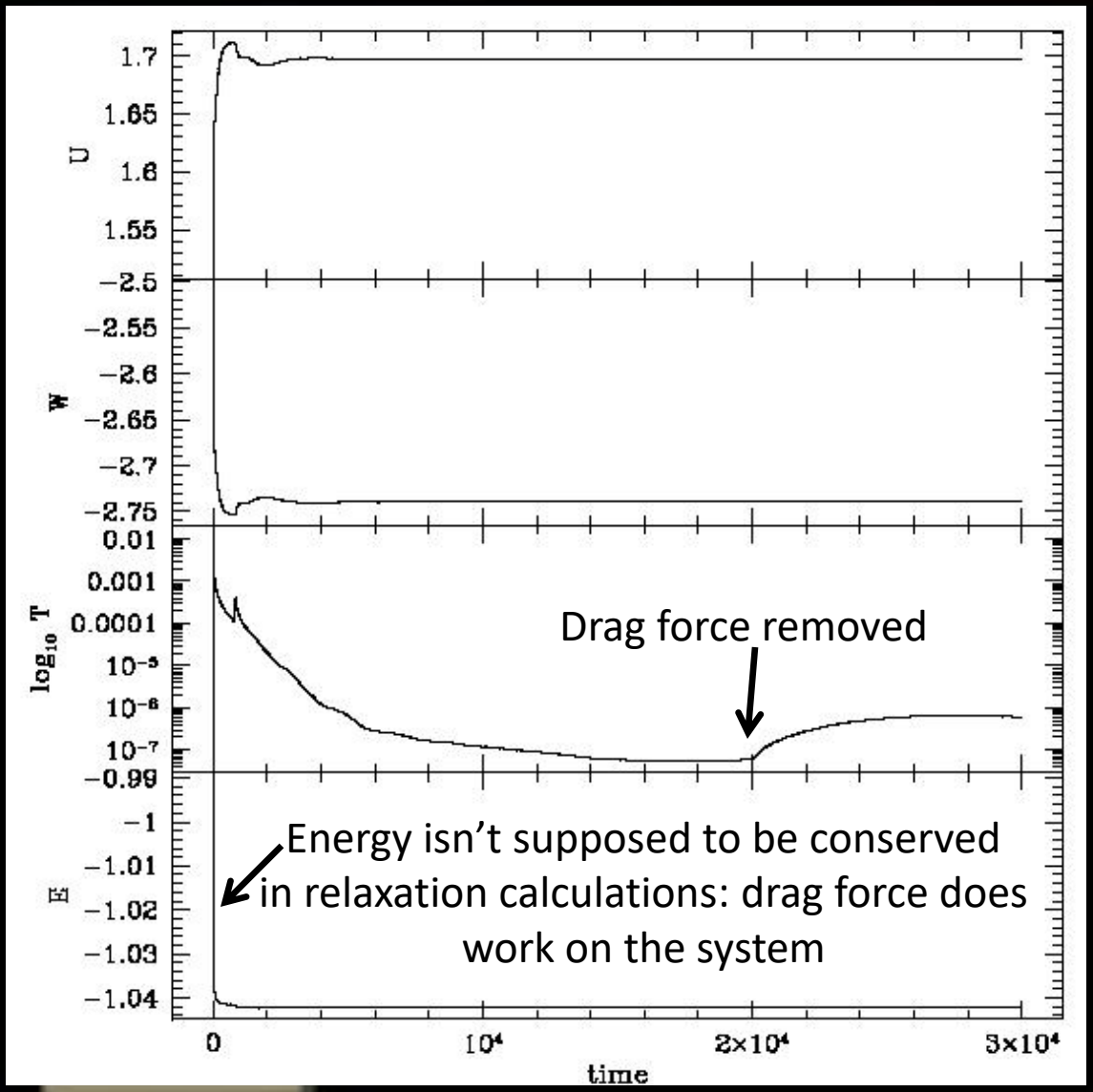
a_{Hydro} (blue) & gravitational acceleration g (green)

Internal energy

Gravitational energy

Kinetic energy (on log scale)

Total energy



Units:
$G=M_{\odot}=R_{\odot}=1$

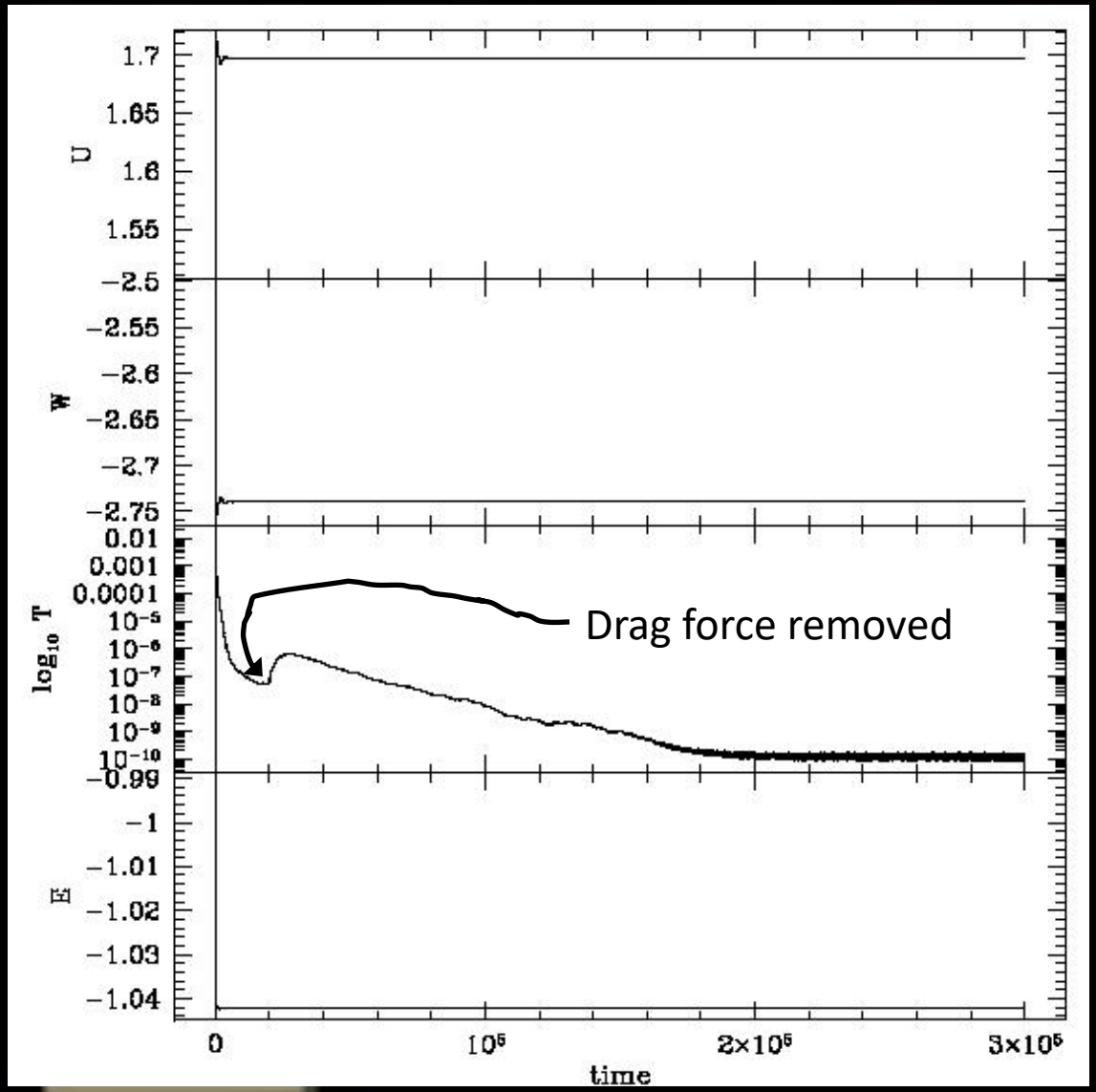
Out to ~30 dynamical times (which took ~30 wallclock minutes with one GPU)

Internal energy

Gravitational energy

Kinetic energy (on log scale)

Total energy

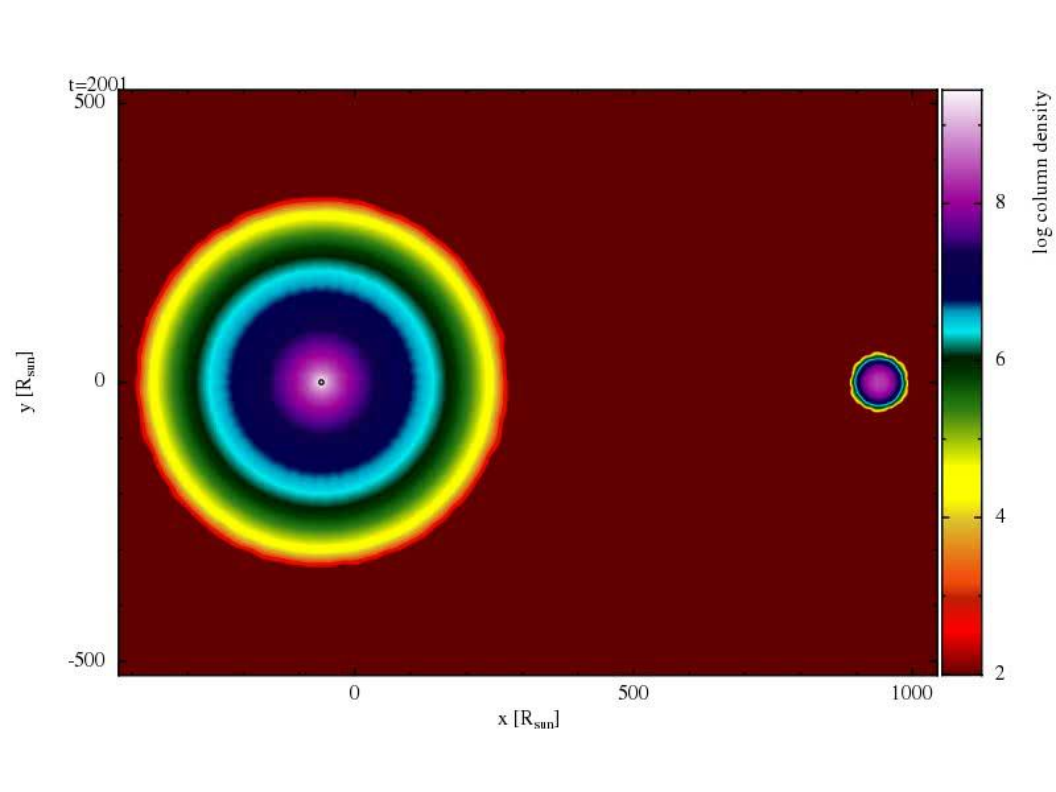


Out to ~300 dynamical times (which took ~4 wallclock hours)

Units:
$G=M_{\odot}=R_{\odot}=1$

How to model a (synchronized) binary
Step 2: Scan binary equilibrium sequences

- Copies of the isolated stars from the relaxation runs are placed in orbit around each other
- Hold in place while oscillations die out, then slowly bring together.
- Angular velocity Ω of corotating frame continually updated to balance centrifugal & physical forces.



Side notes:

- 0.12 s per iteration
- ~3.5 hours wallclock
- Gravity is the bottleneck, taking 80% of the wallclock time.
- Happily, CPUs are working simultaneously with the GPUs for some of that time.

Internal energy

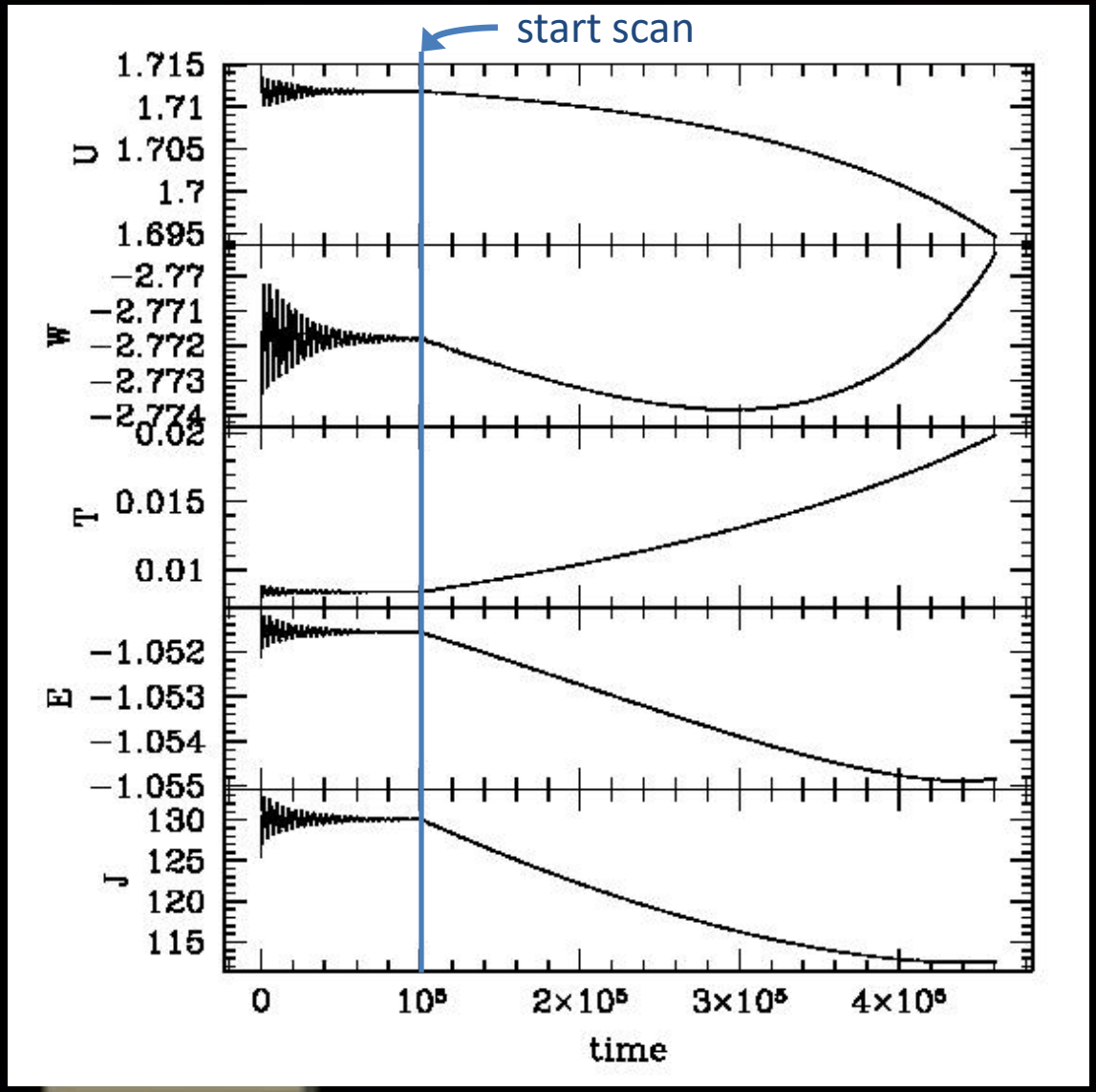
Gravitational energy

Kinetic energy

Total energy

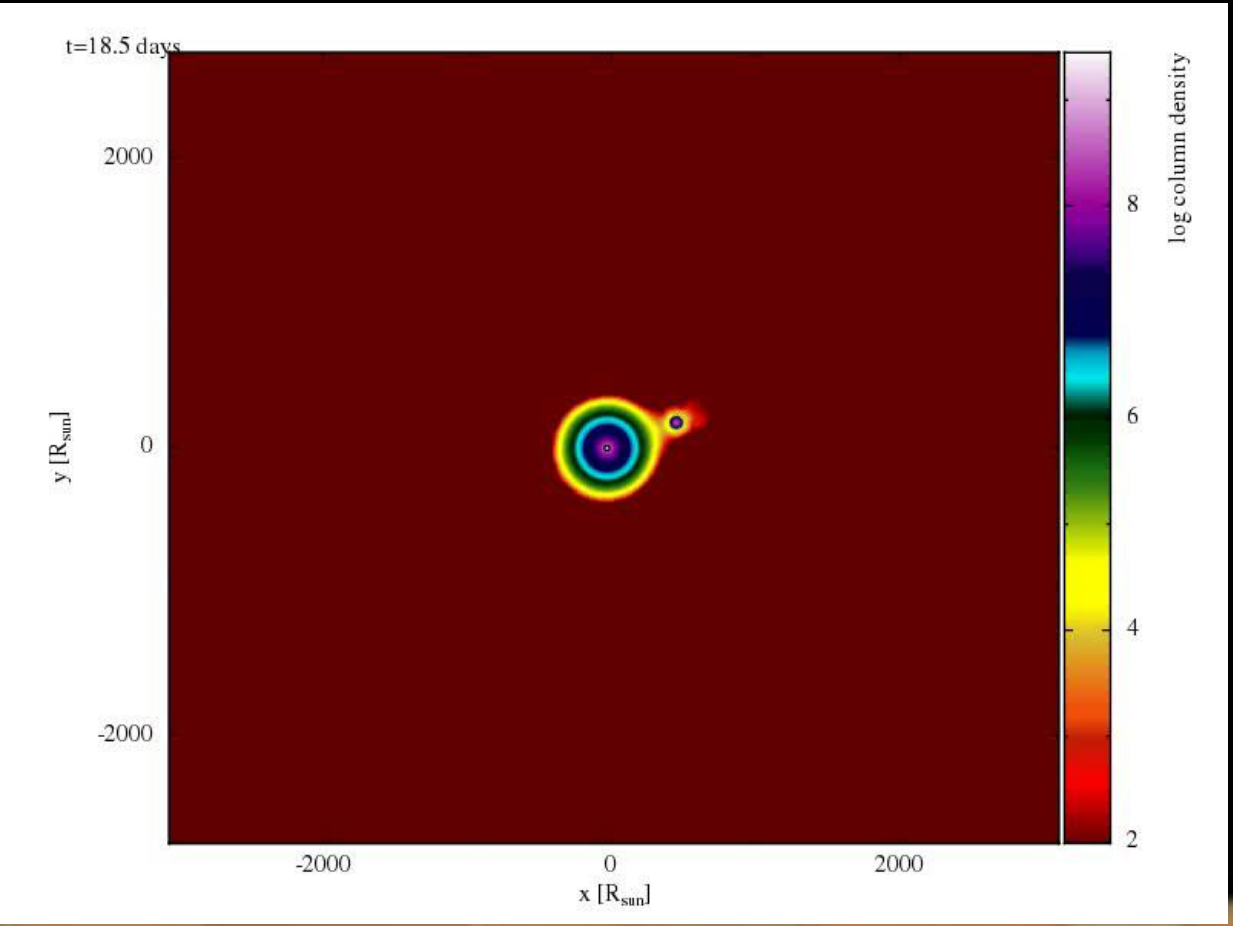
Total angular momentum

Units:
$G=M_{\odot}=R_{\odot}=1$



How to model a (synchronized) binary
Step 3: Perform dynamical runs

- Choose various snapshots from the scanning runs as initial conditions, and allow the system to evolve freely

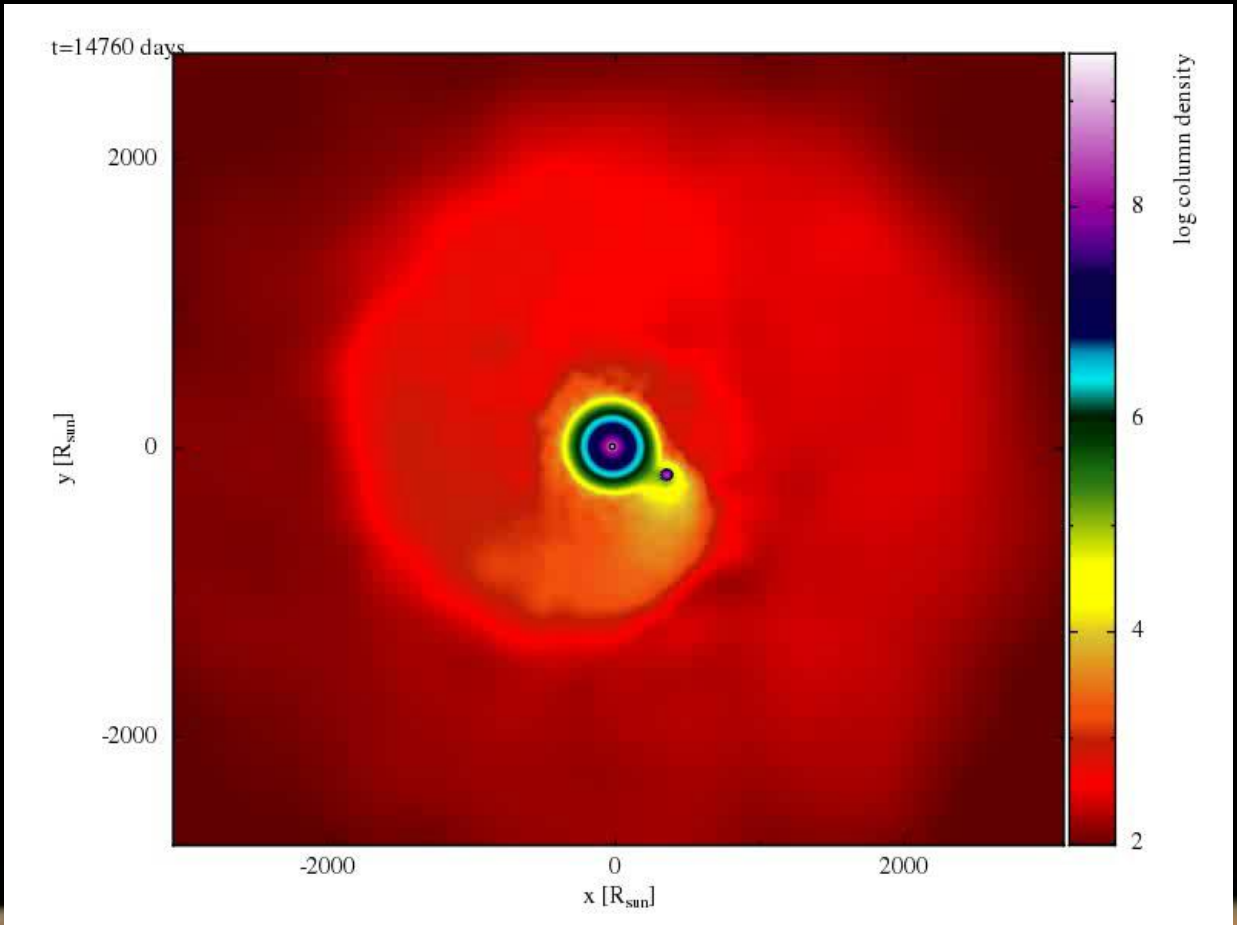


Side notes:

- ~same cost per iteration as binary scan
- ~16 hours wallclock
- 1.1$M_\odot$ ejected

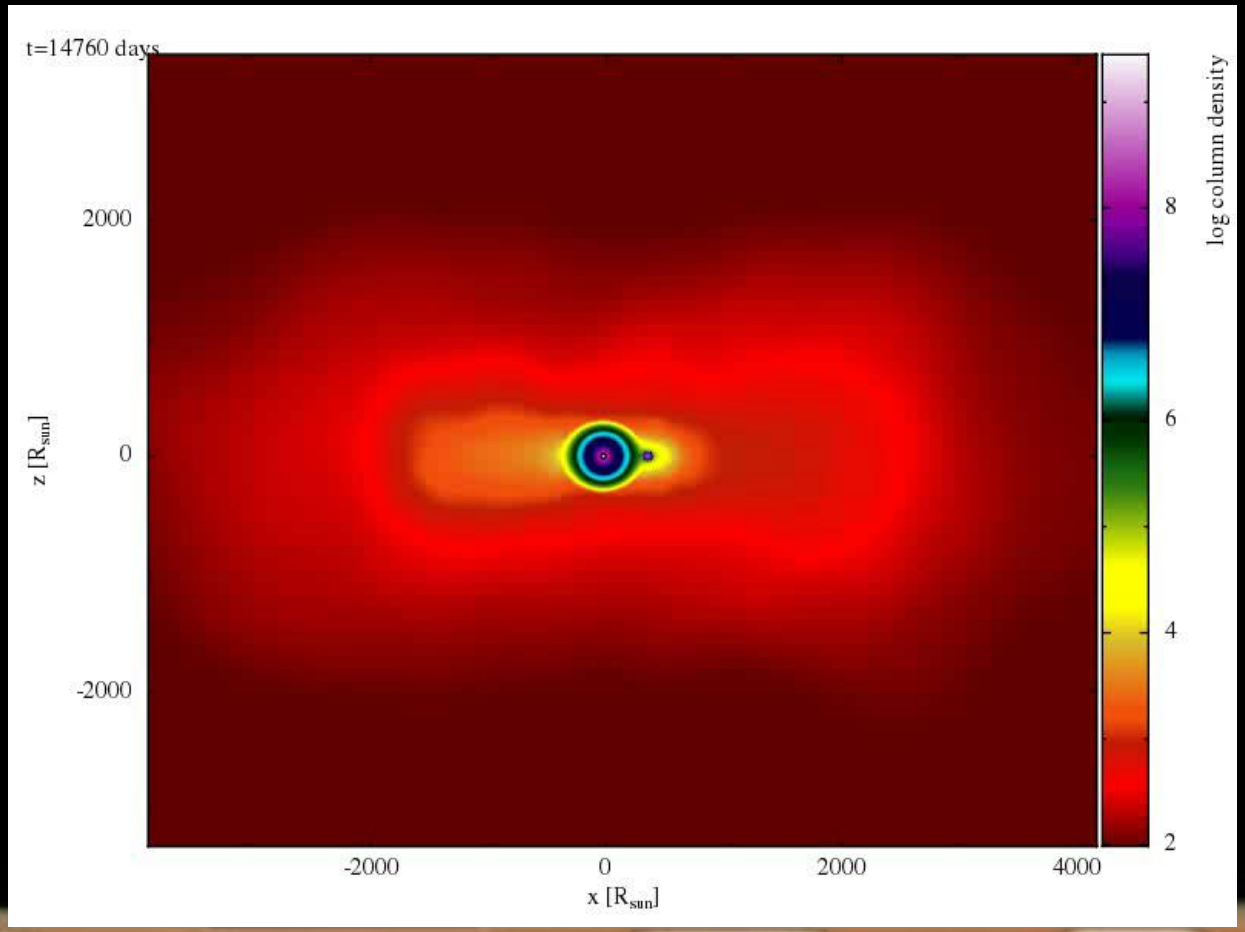
How to model a (synchronized) binary
Step 3: Perform dynamical runs

- Another look at the exciting part…



How to model a (synchronized) binary
Step 3: Perform dynamical runs

- Now a look at column density in *z* vs. *x*



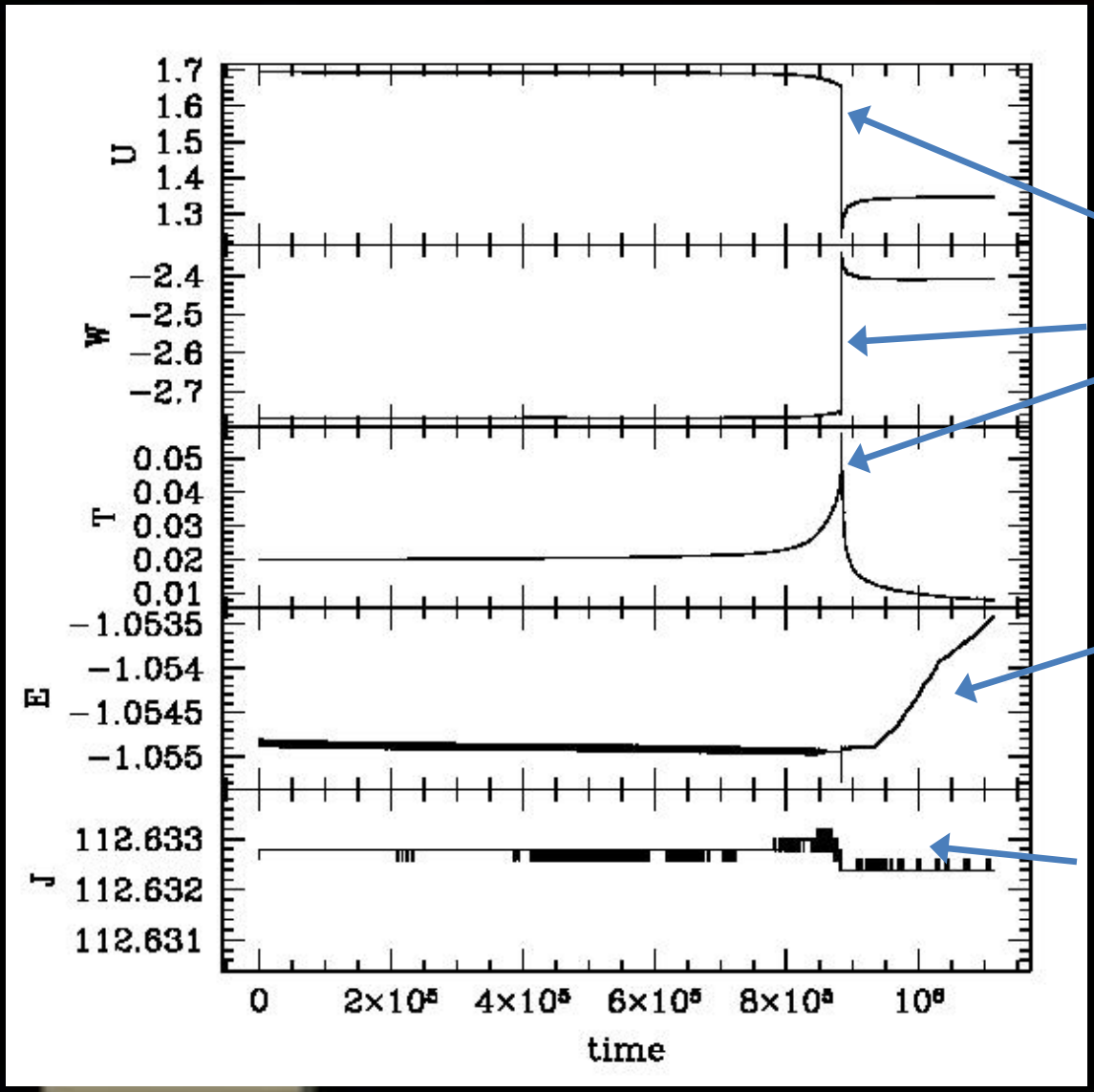
Internal energy

Gravitational energy

Kinetic energy

Total energy

Total angular momentum



~50 orbits before merger

0.14% error (1 part in 700; can be improved w/ smaller timesteps)

1 part in 10^5 error

Units:
$G=M_\odot=R_\odot=1$

Final c.o.m. speed < 2 x 10^{-6}

How to model a (synchronized) binary

Step 4: Complete a convergence study

- As the resolution of the simulation is increased, the effective radius of the parent stars changes.
- Mass transfer rate is sensitive to how the outer layers are modelled.
- Timescale leading up to merger can change (usually lengthen) with increased resolution.

What a bad relaxation can look like…

Previous primary:

$M_{ZAMS} = 16\ M_{\odot}$

Age = 10.006 Myr (base of RG branch)

$M = 15.7\ M_{\odot}$

$R = 292\ R_{\odot}$

Now let's try

$M_{ZAMS} = 20\ M_{\odot}$

Age = 7.75 Myr (base of RG branch)

$M = 19.4\ M_{\odot}$

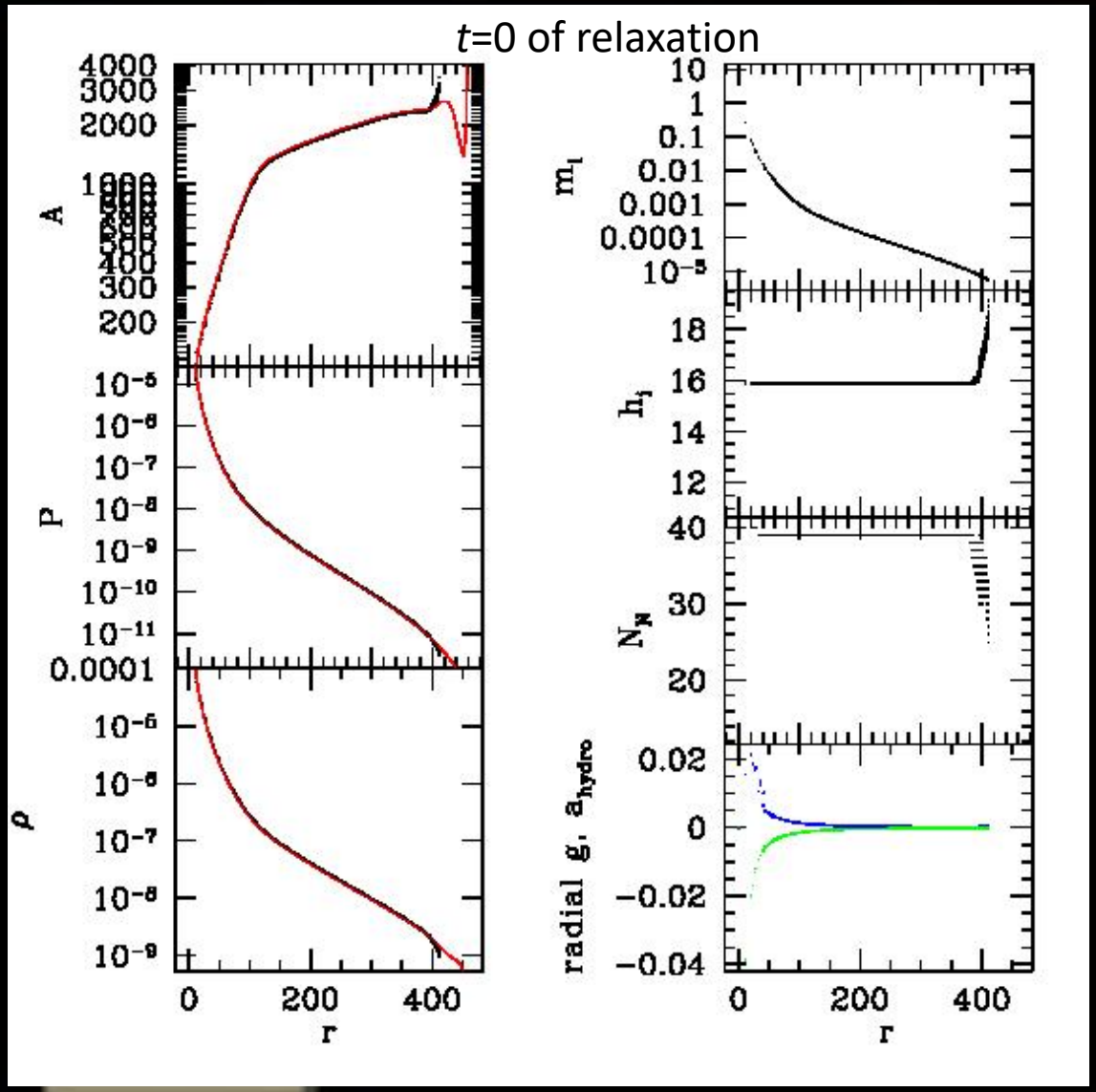
$R = 460\ R_{\odot}$

$P/\rho^{(5/3)}$ (even though EOS is not monatomic ideal gas)

Pressure P

Density ρ

Units:
$G=M_\odot=R_\odot=1$



Particle mass m_i

Smoothing length h_i

Neighbor number N_N

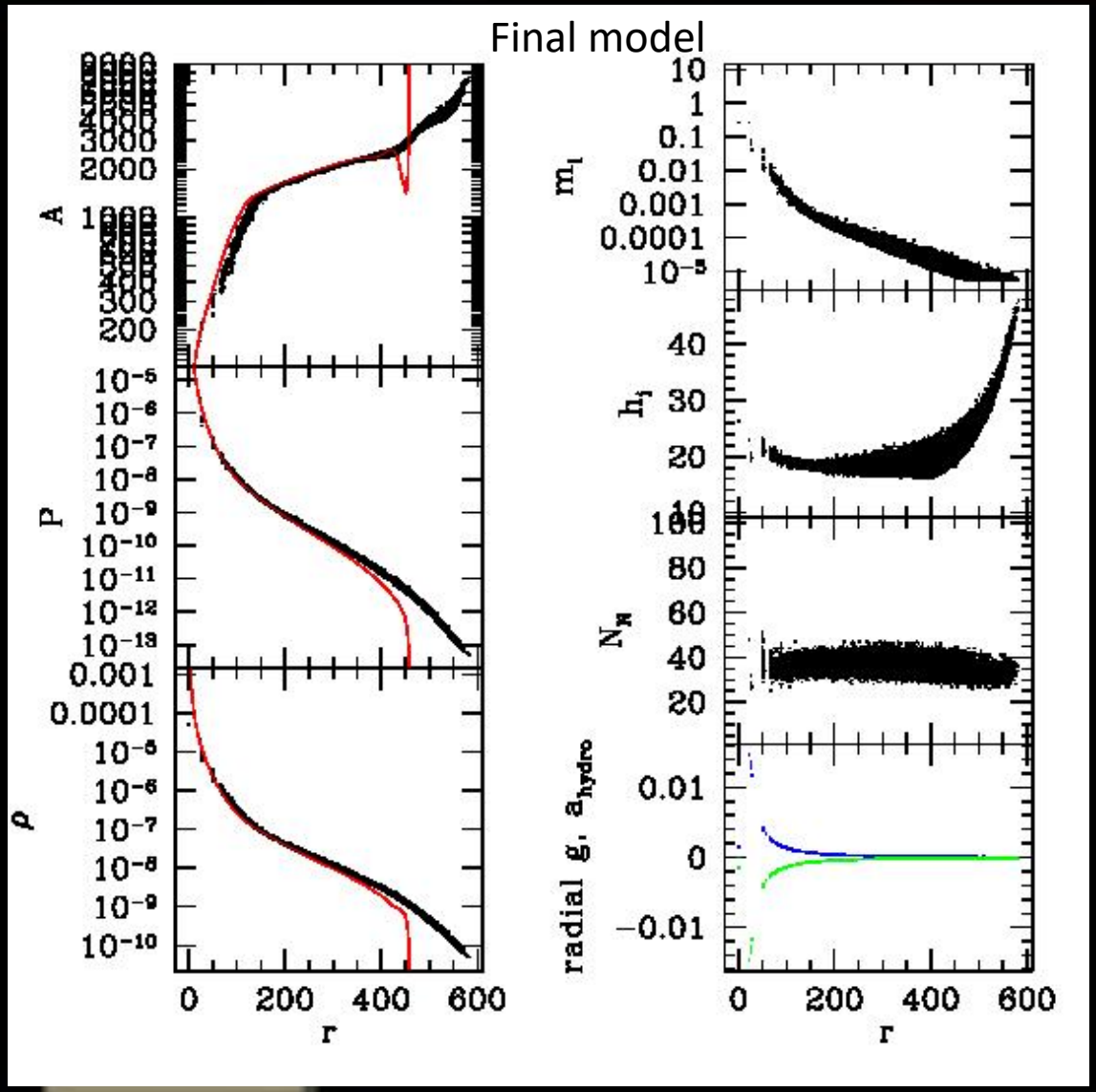
a_{Hydro} (blue) & gravitational acceleration g (green)

$P/\rho^{(5/3)}$ (even though EOS is not monatomic ideal gas)

Pressure P

Density ρ

Units:
$G=M_\odot=R_\odot=1$



Particle mass m_i

Smoothing length h_i

Neighbor number N_N

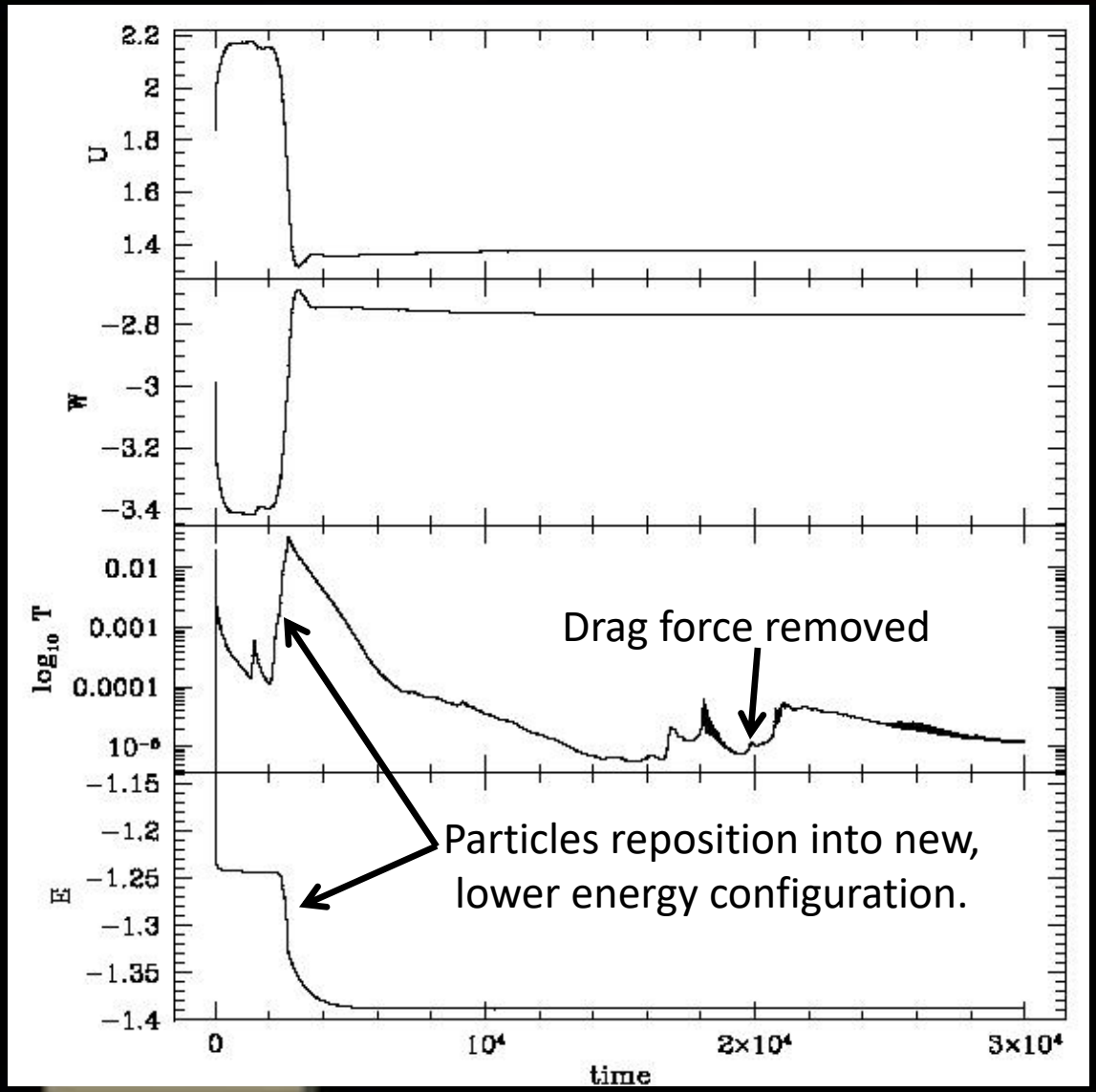
a_{Hydro} (blue) & gravitational acceleration g (green)

Internal energy

Gravitational energy

Kinetic energy (on log scale)

Total energy



Units:
$G=M_{\odot}=R_{\odot}=1$

Things I'd like to improve in *Starsmasher*

- Tabulated EOS used by *Starsmasher* is from MESA, but it assumes all particles have the same composition.
  - This isn't precisely the same EOS used by the stellar evolution code TWIN. (Could this cause relaxation problems?)
  - Generating all 1D models from MESA will help, but still have the problem that *Starsmasher* is currently set up to read in an EOS table of fixed composition.
- Non-synchronized binaries with accurate tidal bulges
  - For mass ratios close to zero (or very large), the primary won't be tidally locked to the secondary
  - Presumably mass ejected when secondary rams through envelope
- Godunov-treatment for shocks to replace classical AV