the `binary` module

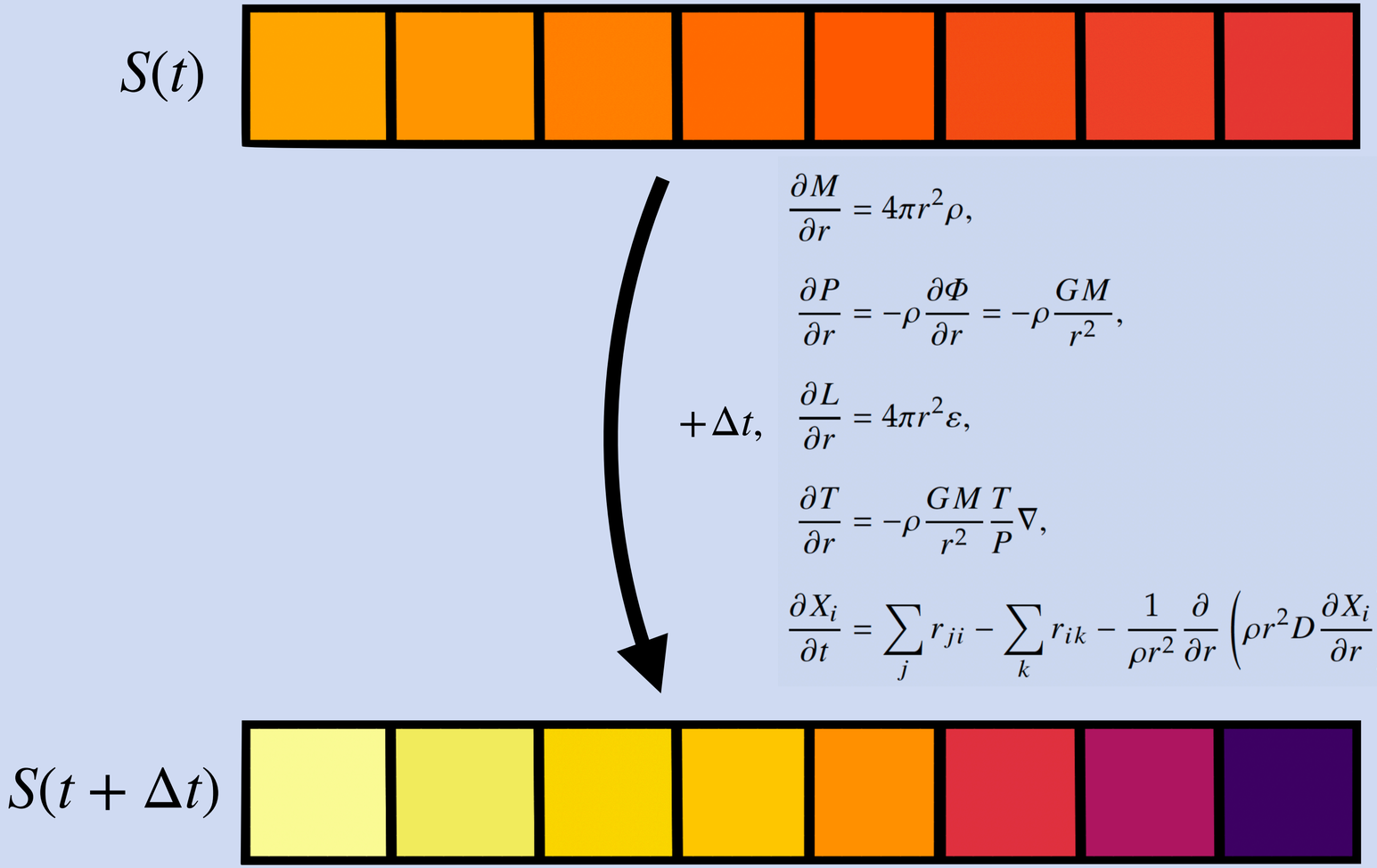
an introduction to binary evolution

Matthias FABRY, 23 July 2025

stellar evolution is fun

model to model

- rotation
- chemical abundances
- overshooting/mixing
- stellar winds
- ...



binary evolution is funner

it takes two

“three out of two stars are in a binary”

binary evolution is funner

it takes two

- cataclysmic variables
- X-ray binaries
- gravitational-wave progenit
- contact binaries
- …

Eva's session!

binary evolution is next generation!

MESA III is featured

talking stars?

evolving two models

$S_1(t)$

$$\frac{\partial M}{\partial r} = 4\pi r^2 \rho,$$

$$\frac{\partial P}{\partial r} = -\rho \frac{\partial \Phi}{\partial r} = -\rho \frac{GM}{r^2},$$

$$\frac{\partial L}{\partial r} = 4\pi r^2 \varepsilon,$$

$$\frac{\partial T}{\partial r} = -\rho \frac{GM}{r^2} \frac{T}{P} \nabla,$$

$$\frac{\partial X_i}{\partial t} = \sum_j r_{ji} - \sum_k r_{ik} - \frac{1}{\rho r^2} \frac{\partial}{\partial r} \left(\rho r^2 D \frac{\partial X_i}{\partial r} \right)$$

$S_1(t + \Delta t)$

$S_2(t)$

$$\frac{\partial M}{\partial r} = 4\pi r^2 \rho,$$

$$\frac{\partial P}{\partial r} = -\rho \frac{\partial \Phi}{\partial r} = -\rho \frac{GM}{r^2},$$

$$\frac{\partial L}{\partial r} = 4\pi r^2 \varepsilon,$$

$$\frac{\partial T}{\partial r} = -\rho \frac{GM}{r^2} \frac{T}{P} \nabla,$$

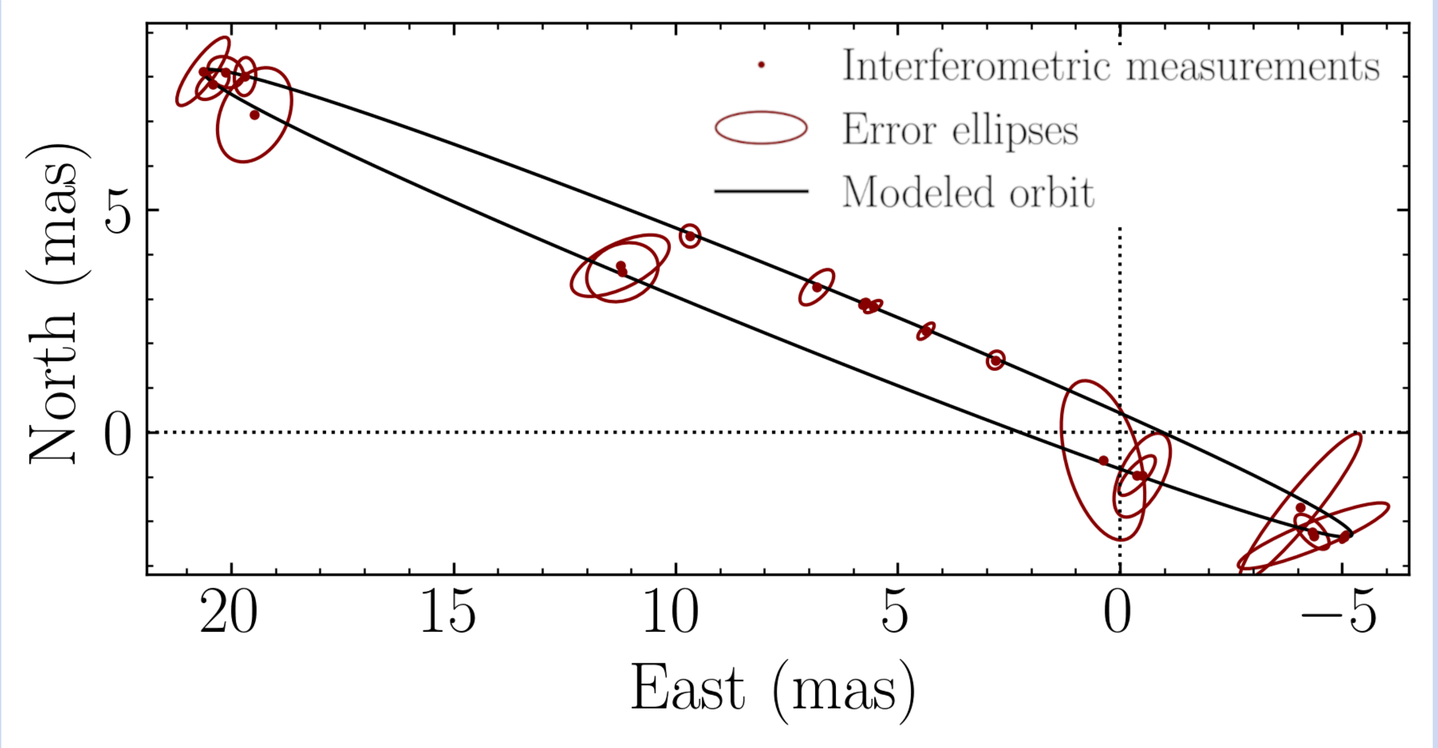
$$\frac{\partial X_i}{\partial t} = \sum_j r_{ji} - \sum_k r_{ik} - \frac{1}{\rho r^2} \frac{\partial}{\partial r} \left(\rho r^2 D \frac{\partial X_i}{\partial r} \right)$$

$S_2(t + \Delta t)$

binary orbits

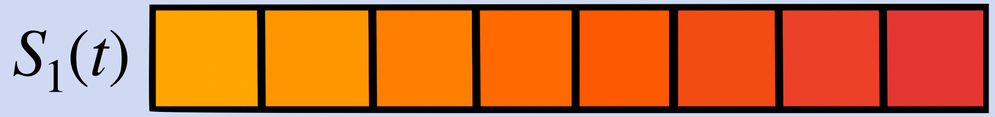
anatomy of a binary

- 4 extra quantities to keep track of:
  - $J_{\rm orb}, M_1, M_2, e \implies a, P_{\rm orb}, E_{\rm orb}$
- mass transfer, wind mass loss
  - $\dot{M}_1, \dot{M}_2$
- angular momentum loss
  - $\dot{J}_{\rm GR}, \dot{J}_{\rm ML}, \dot{J}_{\rm tides}, \dot{J}_{\rm MB}$
- eccentricity changes:
  - $\dot{e}_{\rm tides}, \dot{e}_{\rm third\,body}$



talking stars!

evolving two models



$$\frac{\partial M}{\partial r} = 4\pi r^2 \rho,$$

$$\frac{\partial P}{\partial r} = -\rho \frac{\partial \Phi}{\partial r} = -\rho \frac{GM}{r^2},$$

$$\frac{\partial L}{\partial r} = 4\pi r^2 \varepsilon,$$

$$\frac{\partial T}{\partial r} = -\rho \frac{GM}{r^2} \frac{T}{P} \nabla,$$

$$\frac{\partial X_i}{\partial t} = \sum_j r_{ji} - \sum_k r_{ik} - \frac{1}{\rho r^2} \frac{\partial}{\partial r} \left(\rho r^2 D \frac{\partial X_i}{\partial r} \right)$$

$$\dot{M}_1 = \dot{M}_1(S_1, S_2, P_{\rm orb}, \dots)$$
$$\dot{M}_2 = \dot{M}_2(S_1, S_2, P_{\rm orb}, \dots)$$
$$\dot{J} = \dot{J}(S_1, S_2, P_{\rm orb}, \dots)$$
$$\dot{e} = \dot{e}(S_1, S_2, P_{\rm orb}, \dots)$$

$$\frac{\partial M}{\partial r} = 4\pi r^2 \rho,$$

$$\frac{\partial P}{\partial r} = -\rho \frac{\partial \Phi}{\partial r} = -\rho \frac{GM}{r^2},$$

$$\frac{\partial L}{\partial r} = 4\pi r^2 \varepsilon,$$

$$\frac{\partial T}{\partial r} = -\rho \frac{GM}{r^2} \frac{T}{P} \nabla,$$

$$\frac{\partial X_i}{\partial t} = \sum_j r_{ji} - \sum_k r_{ik} - \frac{1}{\rho r^2} \frac{\partial}{\partial r} \left(\rho r^2 D \frac{\partial X_i}{\partial r} \right)$$

two stars in a box

must be a big box...

- `mesa/star` takes care of evolving the stellar models
- `mesa/binary` makes the stellar models talk

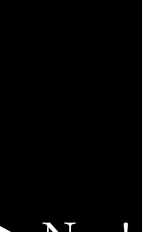
the details

the work directory

cooking up your first binary

- navigate to `$MESA_DIR/binary/work`

```
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:16)-(~/software/mesa-24.08.1/binary/work)> tree -C
.
├── clean
├── inlist
├── inlist1
├── inlist2
├── inlist_project
├── make
│   └── makefile
├── mk
├── re
├── rn
└── src
    ├── binary_run.f90
    ├── run_binary_extras.f90
    └── run_star_extras.f90

3 directories, 12 files
```



the work directory

cooking up your first binary

- there's `binary_job` just like `star_job`
  - point to individual star inlists
  - define high-level, non-changing controls
- there's `binary_controls` like `controls`
  - initial (binary) state
  - mass-transfer controls
  - angular momentum controls
  - (binary) timestep controls
- see https://docs.mesastar.org/en/24.08.1/reference.html for all controls.

```
&binary_job

   inlist_names(1) = 'inlist1'
   inlist_names(2) = 'inlist2'

   evolve_both_stars = .false.

/ ! end of binary_job namelist

&binary_controls
   !initial conditions
   m1 = 1.0d0 ! donor mass in Msun
   m2 = 1.4d0 ! companion mass in Msun
   initial_period_in_days = 2d0

   !mass transfer controls
   limit_retention_by_mdot_edd = .true.
   max_tries_to_achieve = 20
   !timestep controls
   fr = 1d-2

/ ! end of binary_controls namelist
```

the work directory

cooking up your first binary

- there's `run_binary_extras`
  - define custom controls for various physics
  - define extra columns to be recorded in `binary_history`
  - perform calculations at start/end of step/evolution

```
module run_binary_extras

   contains

   subroutine extras_binary_controls(binary_id, ierr)
      integer :: binary_id
      integer, intent(out) :: ierr
      type (binary_info), pointer :: b
      ierr = 0
      call binary_ptr(binary_id, b, ierr)
      if (ierr /= 0) then
         write(*,*) 'failed in binary_ptr'
         return
      end if
…
   end subroutine extras_binary_controls
…
   integer function how_many_extra_binary_history_header_items(binary_id)
…
   subroutine data_for_extra_binary_history_header_items( &
           binary_id, n, names, vals, ierr)
…
   integer function how_many_extra_binary_history_columns(binary_id)
…
   subroutine data_for_extra_binary_history_columns(binary_id,&
                       n, names, vals, ierr)
…
   integer function extras_binary_startup(binary_id,restart,ierr)
…
   integer function extras_binary_start_step(binary_id,ierr)
…
   integer function extras_binary_check_model(binary_id)
…
   ! returns either keep_going or terminate.
   ! note: cannot request retry; extras_check_model can do that.
   integer function extras_binary_finish_step(binary_id)
…
   subroutine extras_binary_after_evolve(binary_id, ierr)
end module run_binary_extras
```

the `binary_info` type

a binary's bookshelf

- much like `star_info`, we need a container to keep all the information relevant to compute binary evolution.
  - *ecce!* `type (binary_info), pointer :: b`
- now we can access any field of `b` defined in `binary_controls, binary_data`, eg.:
  - `b% m(1)`
  - `b% mtransfer_rate`
  - `b% period`
  - `b% do_tidal_circ`
- many controls are set in the inlist at the start of the simulation

control flow

control flow

testing the currents

- copy the work directory to your place of choice (it's useful to have a `projects/` folder somewhere) and `cd` to it.

```
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:26)-(~/software/mesa-24.08.1/binary/work)> cp -r * ~/projects/mesa_school_2025/first_binary
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:26)-(~/software/mesa-24.08.1/binary/work)> cd ~/projects/mesa_school_2025/first_binary
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:28)-(~/projects/mesa_school_2025/first_binary)>
```

- write some `write` statements in the `extras` routines:
  - binary: `binary_start_step, binary_check_model, binary_finish_step`
  - star (first copy contents of `$MESA_DIR/star/job/standard_run_star_extras.inc`): `start_step, check_model, finish_step`

control flow

testing the currents

- e.g. `binary_start_step`:
  insert as very first statement
  after declaration of all variables!

```
integer function extras_binary_start_step(binary_id,ierr)
   type (binary_info), pointer :: b
   integer, intent(in) :: binary_id
   integer, intent(out) :: ierr

   write(*, *) "doing binary_start_step"

   extras_binary_start_step = keep_going
   call binary_ptr(binary_id, b, ierr)
   if (ierr /= 0) then ! failure in  binary_ptr
      return
   end if
end function  extras_binary_start_step
```

- e.g. star `start_step`:
  (remember we potentially have
  two stars to track! each with a
  different `id`)
  (`standard_run_star_extras.inc`
  is located in `star/job/`)

```
integer function extras_start_step(id)
   integer, intent(in) :: id
   integer :: ierr
   type (star_info), pointer :: s

   write(*, *) "doing star", id, "start_step"

   ierr = 0
   call star_ptr(id, s, ierr)
   if (ierr /= 0) return
   extras_start_step = 0
end function extras_start_step
```

control flow

testing the currents

- let's run for ten steps to see the control flow
  - `./mk`, `./rn`. `ctrl+C` to stop

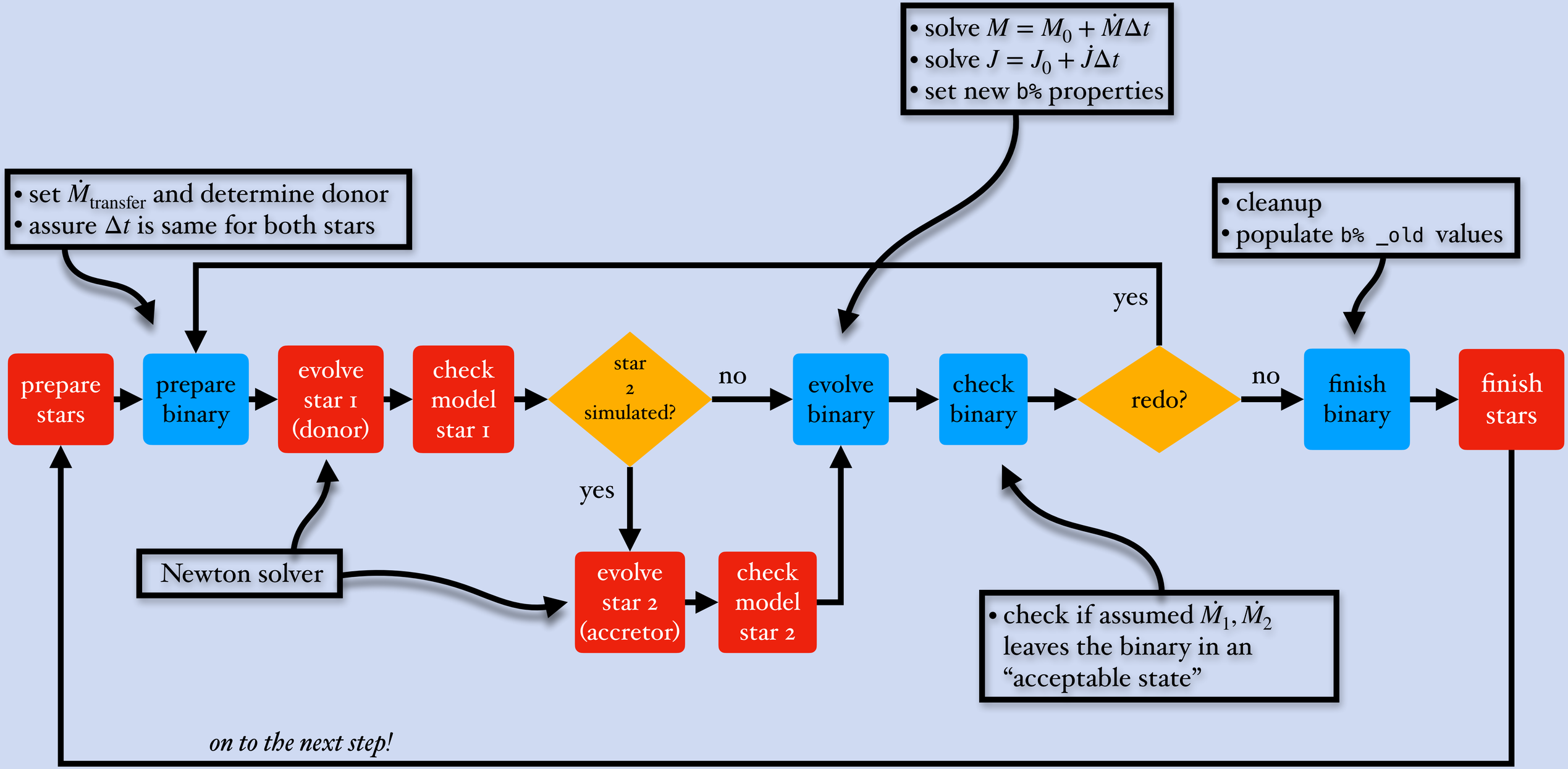
```
 doing star           1 start_step
 doing binary_start_step
 doing star           1 check_model
          2   7.133759   5615.044  -0.152227  -0.152227   1.000000   1.000000   0.696983   0.004061   0.280000  -1.726624   1937      0
 5.0792E+00   7.133759  -0.050903 -47.003331  -1.764749 -99.000000   0.000000   0.282581   0.009380   0.020000   0.085689      2
 2.2000E+05   1.905413  -0.149700 -99.000000 -99.000000  -6.704739   0.000000   0.000833   0.002085   0.020436  0.000E+00  max increase

 doing binary_check_model
 doing binary_finish_step
bin       2   2.400000   8.943924   1.999999  0.000E+00   1.400000          2          1  0.000E+00   1.000000  1.633E+52 -2.261E+32  0.000E+00
   5.079181   1.000000   0.889400   0.000000  0.000E+00 131.978495   3.132084 -7.160E-01  0.000E+00  6.357E-08  0.000E+00 -2.261E+32  0.000E+00
 2.2000E+05   1.400000   0.000000   0.000000 -2.969E+47  94.270353   3.652301 -1.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00          1

 doing star           1 finish_step
 doing star           1 start_step
 doing binary_start_step
 doing star           1 check_model
          3   7.133759   5615.132  -0.152360  -0.152360   1.000000   1.000000   0.696975   0.004067   0.280000  -1.725934   1249      0
 5.1584E+00   7.133759  -0.050873 -47.006384  -1.764965 -99.000000   0.000000   0.282589   0.009380   0.020000   0.085708      2
 3.6400E+05   1.905698  -0.149613 -99.000000 -99.000000  -6.704793   0.000000   0.000827   0.002085   0.020437  0.000E+00  max increase

 doing binary_check_model
 doing binary_finish_step
bin       3   2.400000   8.943923   1.999999  0.000E+00   1.400000          2          1  0.000E+00   1.000000  1.633E+52 -2.261E+32  0.000E+00
   5.158362   1.000000   0.889460   0.000000  0.000E+00 131.978503   3.132083 -7.160E-01  0.000E+00  6.357E-08  0.000E+00 -2.261E+32  0.000E+00
 3.6400E+05   1.400000   0.000000   0.000000 -2.969E+47  94.270359   3.652300 -1.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00          1

 doing star           1 finish_step
```

control flow

testing the currents

- change `evolve_both_stars = .true.`
  - `./rn`

```
 doing star           1 start_step
 doing star           2 start_step
 doing binary_start_step
 Default mdot_edd calculation cannot be used when evolving both stars
 Maybe you want to set limit_retention_by_mdot_edd=.false. in binary_controls?
 Setting mdot_edd to zero
 doing star           1 check_model
  1       8   7.133787   5615.134  -0.152359  -0.152359   1.000000   1.000000   0.696975   0.004067   0.280000  -1.725939    802      1
 4.8879E+00   7.133787  -0.050874 -47.004363  -1.764912 -99.000000   0.000000   0.282588   0.009380   0.020000   0.085705      2
 3.5572E+05   1.905738  -0.149613 -99.000000 -99.000000  -6.704793   0.000000   0.000827   0.002085   0.020437  0.000E+00  max increase

 doing star           2 check_model
  2       8   7.241946   6678.421   0.540135   0.540135   1.400000   1.400000   0.697209   0.004993   0.280000  -2.104765    885      1
 4.8879E+00   7.241946   0.145296 -41.543815  -0.990817 -99.000000   0.000000   0.282226   0.009365   0.020000   0.067129      3
 3.5572E+05   1.911689   0.543982 -99.000000 -99.000000  -7.303330   0.000000   0.000045   0.002085   0.020565  0.000E+00  max increase

 doing binary_check_model
 doing binary_finish_step
bin       8   2.400000   8.943923   1.999999  0.000E+00   1.400000          0          1  0.000E+00   1.000000  1.633E+52 -2.261E+32  0.000E+00
   4.887919   1.000000   0.889460   0.000000  0.000E+00 131.978503   3.132083 -7.160E-01  0.000E+00  0.000E+00  0.000E+00 -2.261E+32  0.000E+00
 3.5572E+05   1.400000   1.397322   0.000000 -2.969E+47  94.270359   3.652300 -6.174E-01  0.000E+00  0.000E+00  0.000E+00  0.000E+00          1

 doing star           1 finish_step
 doing star           2 finish_step
```


• solve $M = M_0 + \dot{M}\Delta t$
• solve $J = J_0 + \dot{J}\Delta t$
• set new b% properties
• set $\dot{M}_{\rm transfer}$ and determine donor
• assure Δt is same for both stars
• cleanup
• populate b% _old values
yes
prepare stars
prepare binary
evolve star 1 (donor)
check model star 1
star 2 simulated?
no
evolve binary
check binary
redo?
no
finish binary
finish stars
yes
Newton solver
evolve star 2 (accretor)
check model star 2
• check if assumed $\dot{M}_1, \dot{M}_2$ leaves the binary in an "acceptable state"
on to the next step!

custom stopping condition

to know when to stop...

- exercise!
  - turn `evolve_both_stars` back to `.false.`, set the period to 0.5 days, and turn off magnetic braking with `do_jdot_mb = .false.` in `&binary_controls`.
  - now stop the simulation once the star approaches the Roche Lobe (say, when it is 0.95 times the RL size)

```
if (b% r(1) >= 0.95 * b% rl(1)) then
   write(*, *) "star is approaching the Roche Lobe! Mass transfer will ensue, I'm stopping the run here..."
   extras_binary_finish_step = terminate
end if
```

the `binary` folder

the binary folder

where the magic happens

```
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:12)-(~/software/mesa-24.08.1/binary/private)> tree -CL 1
.
├── binary_ce.f90
├── binary_ctrls_io.f90
├── binary_do_one_utils.f90
├── binary_edot.f90
├── binary_evolve.f90
├── binary_history.f90
├── binary_history_specs.f90
├── binary_irradiation.f90
├── binary_jdot.f90
├── binary_job_controls.inc
├── binary_job_ctrls_io.f90
├── binary_mdot.f90
├── binary_photos.f90
├── binary_private_def.f90
├── binary_tides.f90
├── binary_timestep.f90
├── binary_utils.f90
├── binary_wind.f90
├── pgbinary_ctrls_io.f90
├── pgbinary_full.f90
├── pgbinary_grid.f90
├── pgbinary_hist_track.f90
├── pgbinary_history_panels.f90
├── pgbinary_lib.f90
├── pgbinary_orbit.f90
├── pgbinary_star.f90
├── pgbinary_stub.f90
├── pgbinary_summary.f90
├── pgbinary_summary_history.f90
├── pgbinary_support.f90
└── run_binary_support.f90

1 directory, 31 files
```

Routines for angular momentum losses

Routines for adapting the timestep

Binary plotting library

Main file with binary evolution control flow

the `binary` folder

where the magic happens

- situation: you want to know how magnetic braking is implemented. `jdot_mb` is the relevant control, where is the `subroutine`?

- `grep -rn` is your friend!

```
(base) (MESAr24.08.1) (matthiasf@tejat)-(25-04-15 14:12)-(~/software/mesa-24.08.1/binary/private)> grep -rn "subroutine.*jdot_mb"
./binary_jdot.f90:208:      subroutine check_jdot_mb_conditions(b, s, apply_jdot_mb, qconv_env)
./binary_jdot.f90:253:     end subroutine check_jdot_mb_conditions
./binary_jdot.f90:255:      subroutine default_jdot_mb(binary_id, ierr)
./binary_jdot.f90:336:      end subroutine default_jdot_mb
```

- Inside `binary_jdot.f90`:

```
283: b% jdot_mb = -3.8d-30*b% m(b% d_i)*rsun4* &
                pow(min(b% r(b% d_i),b% rl(b% d_i))/rsun,b% magnetic_braking_gamma)* &
                two_pi_div_p3*jdot_scale
```

- From Rappaport et al. (1983): $\dot{J} = -3.8 \cdot 10^{-30} M_d R_\odot^4 \left(\frac{R}{R_\odot}\right)^\gamma \frac{2\pi}{p_{\rm orb}^3} \mathrm{dyn\,cm}$

the binary folder

where the magic happens

Exercise!

- is angular momentum loss from gravitational radiation correctly implemented?

$$\frac{\dot{J}}{J} = -\frac{32}{5}\frac{G^3}{c^5 a^4} M_1 M_2 \left(M_1 + M_2\right)$$

Solution:
```
grep -rn "subroutine.*jdot_gr"

binary_jdot.f90:115
```

- verify the implementation of the mass-transfer rate determination from Ritter (1988):

$$\dot{M} = -\frac{2\pi}{\sqrt{e}}\frac{c_s^3 R_{\mathrm{RL}}^3}{GM}\rho_{\mathrm{photosphere}} F(q) \exp\left(-\frac{R_{\mathrm{RL}} - R}{\frac{H_P}{\gamma(q)}}\right)$$

Solution:
```
grep -rn “subroutine.*ritter”

binary_mdot.f90:711-730
```

binary hooks

aargh matey!

- just like in `star`, there are many hooks built into `mesa/binary` that allow for custom physics implementation
- the `binary/other/mod….f90` files contain the "null" functions for these hooks, and tell you the input/output for each function.
- *eg.*: you can implement your own prescription for magnetic braking, mass transfer rate, etc.

the test_suite and pgbinary

the `test_suite`

recipes for more refined tastes

- used to test `mesa/binary` functionality
- templates for some science cases
- NOT SCIENCE-GRADE by itself (like `star/test_suite`)
  - limited resolution
  - ignored physics

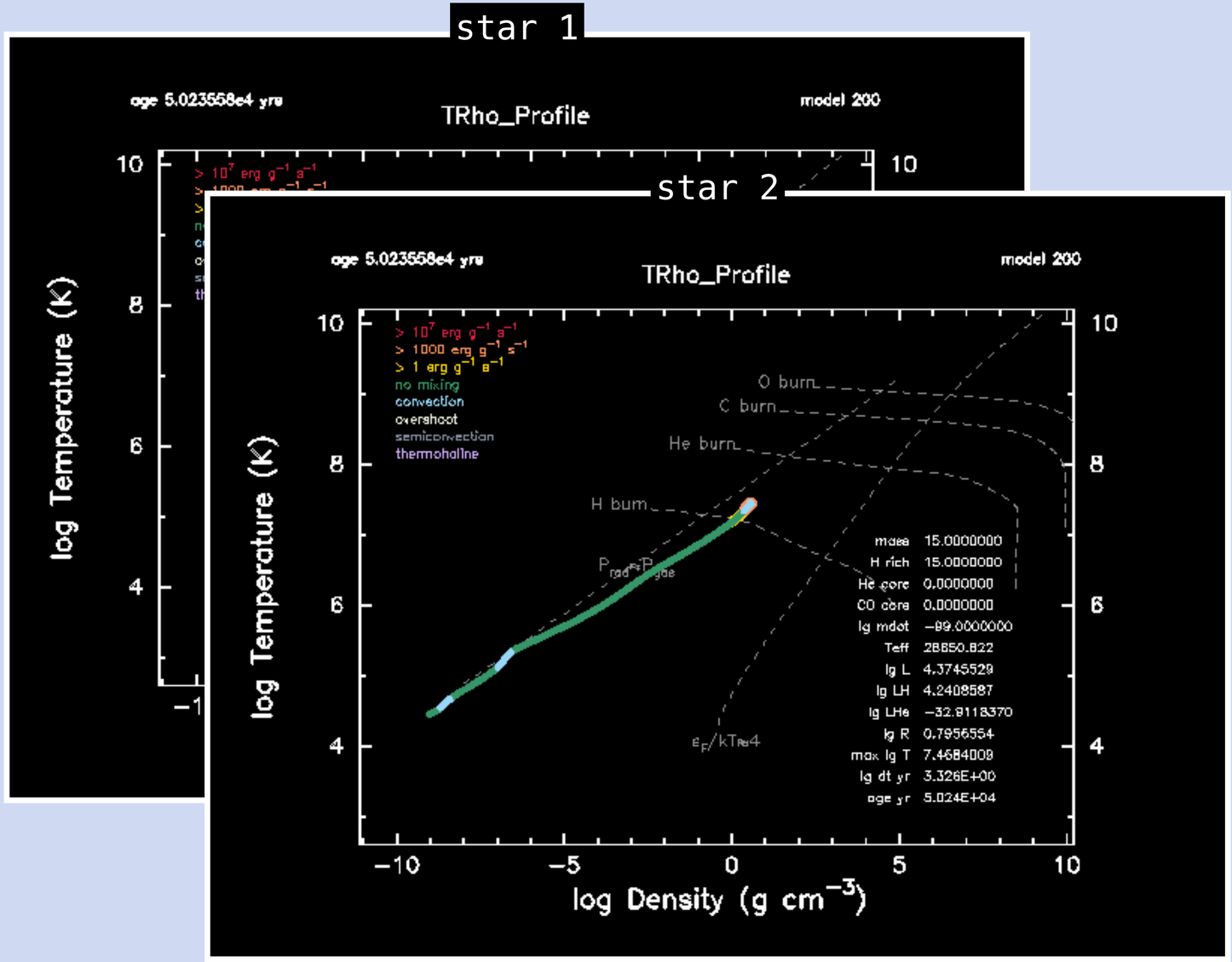
```
(base) (MESAr24.08.1) (matthiasf@tejat)-
(25-04-16 14:33)-(~/software/mesa-24.08.1/
binary/test_suite)> tree -CL 1
.
├── count_tests
├── do1_test_source
├── double_bh
├── each_test_clean
├── each_test_compile
├── each_test_run
├── evolve_both_stars
├── jdot_gr_check
├── jdot_ls_check
├── jdot_ml_check
├── list_tests
├── report
├── star_plus_point_mass
├── star_plus_point_mass_explicit_mdot
└── wind_fed_bhhmxb

9 directories, 7 files
```

pgbinary

I see, I see, a little binary

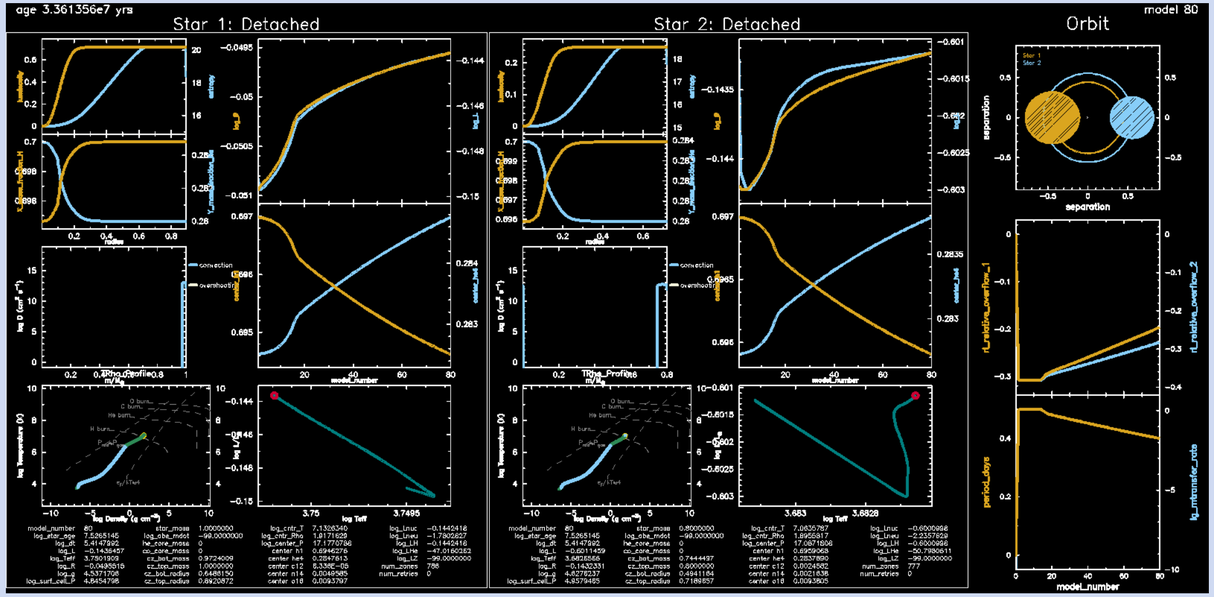
- visualization important for:
  - verification
  - education
  - publication (but plz use something else than PGPLOT of MESA to make publication-grade plots, ok thx.)

- previously:
  - 2 `pgstar` windows
  - doubled binary info (*eg* `b% period`)



pgbinary

I see, I see, a little binary



- now:
  - 1 `pgbinary` window
  - to-scale representation of system
- test it!
  - copy `evolve_both_stars` into your projects folder
  - uncomment `pgbinary_flag = .true.`
  - set the following timestep controls:

```
fr = 5d-2
fr_limit = 1d-1
fj = 5d-2
```

  - add `pause_before_terminate = .true.` in `inlist1`
  - un the simulation: `./mk; ./rn`
  - you might want to adapt the window size to fit your screen
  - you might want to adapt text scaling (both in `pgbinary` *and* `pgstar`)

→

```
Grid1_win_width = 19
Grid1_win_aspect_ratio = 0.49
```

→

```
Grid1_txt_scale_factor(1) = 0.9
```

pgbinary

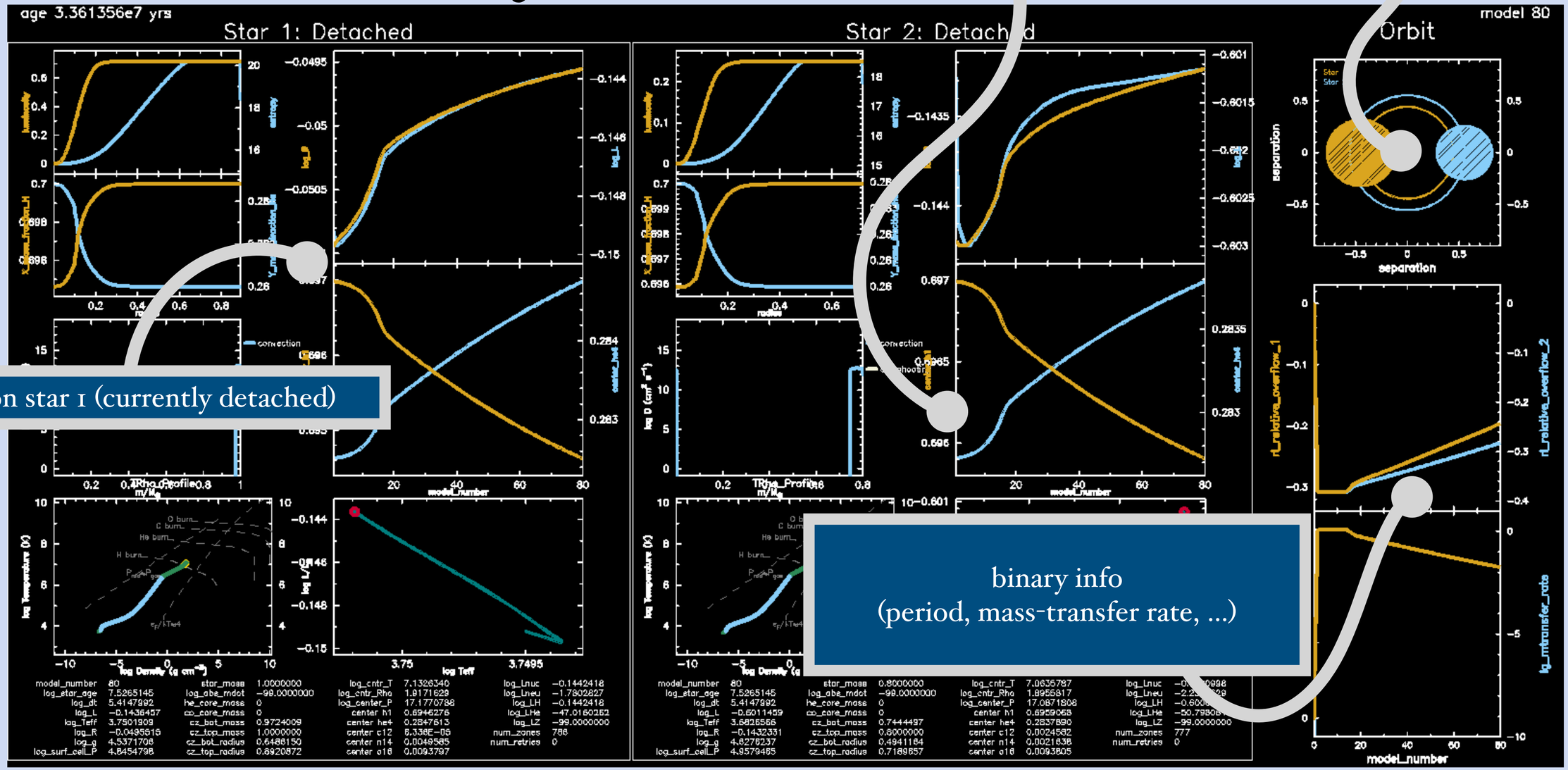
I see, I see, a little binary

crude representation of the system

info on star 2 (currently detached)

info on star 1 (currently detached)

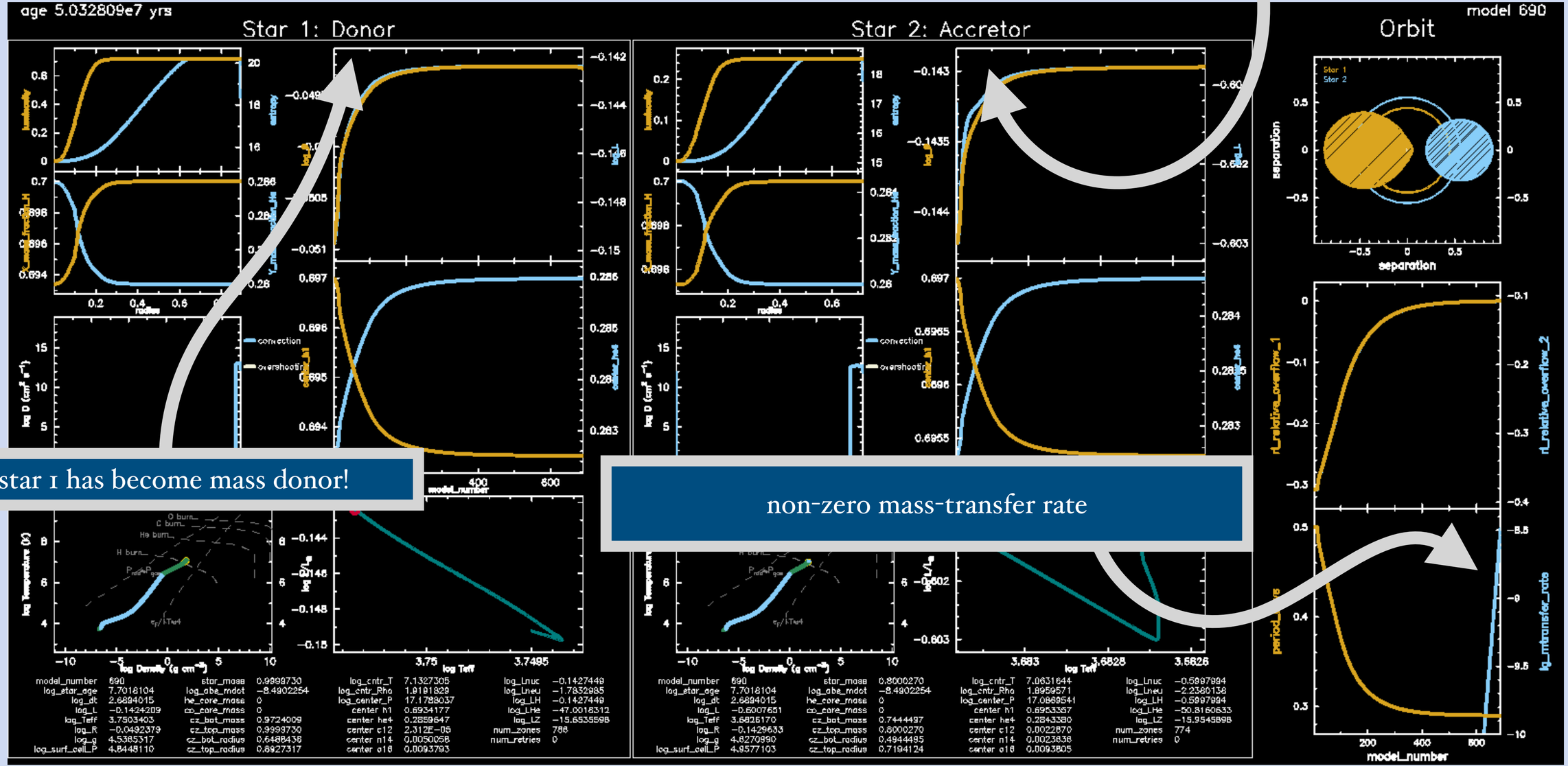
binary info
(period, mass-transfer rate, ...)



pgbinary

I see, I see, a little talking binary!

star 2 is the accretor



star 1 has become mass donor!

non-zero mass-transfer rate

pgbinary structure

I see, I see, a little binary...

analyse inlist_pgbinary:

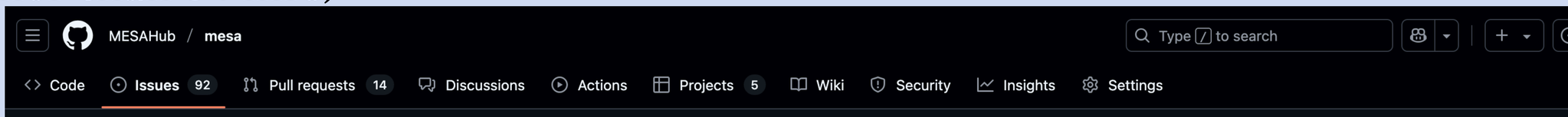
- pgbinary plots 2 windows:
  - `Orbit` (a nice representation of the stars in the system, to scale)
  - `Grid1` (which in turn plots several panels:)
    - `Star1` (this in turn commands to plot `Grid2` for `s1%` using `pgstar` settings)
      - see `inlist_pgstar` for subplots of `Grid2`
    - `Star2` (also commands to plot `Grid2`, but for `s2%` of course)
      - same
    - `Orbit`
    - `History_panels` (functions just like `pgstar` `history_panels`, only using `b%` info rather than `s%` info)

pgbinary structure

I see, I see, a little binary...

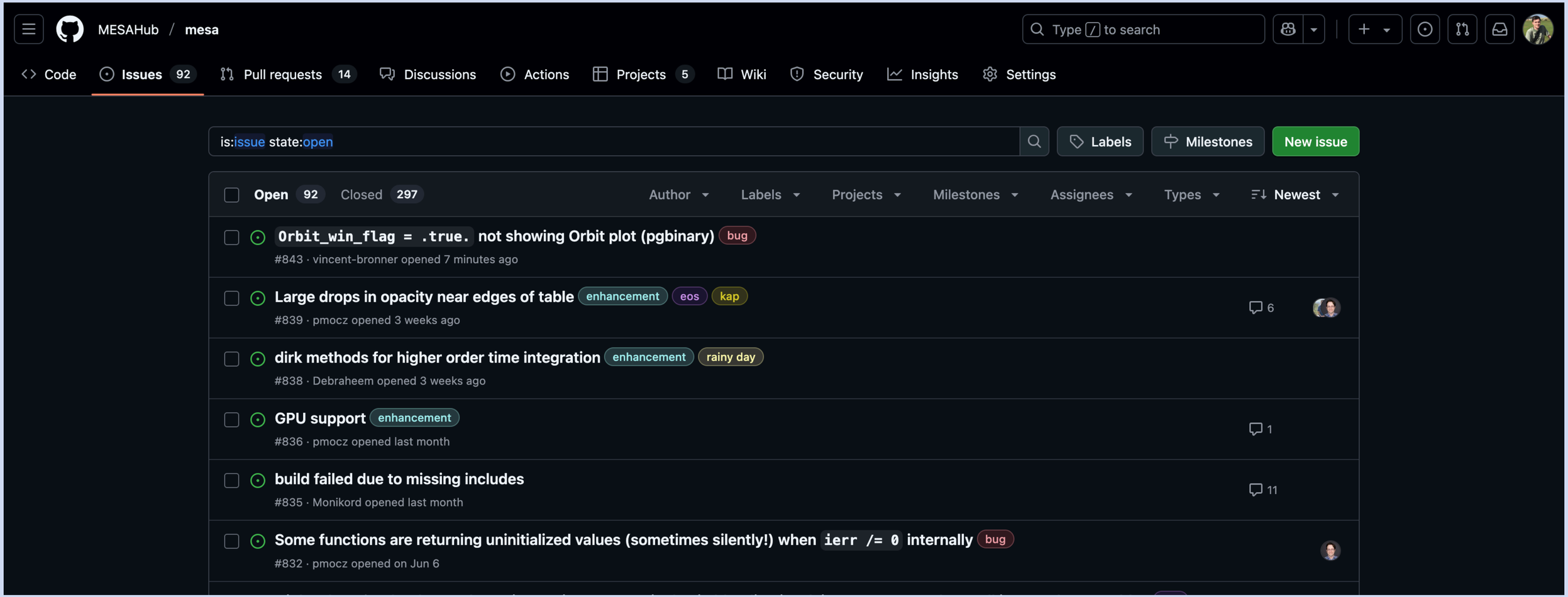
analyse inlist_pgbinary:

- pgbinary plots 2 windows:
  - ~~`Orbit` (a nice representation of the stars in the system, to scale)~~
  - `Grid1` (which in turn plots several panels:)
    - `Star1` (this in turn commands to plot `Grid2` for `s1%` using `pgstar` settings)
      - see `inlist_pgstar` for subplots of `Grid2`
    - `Star2` (also commands to plot `Grid2`, but for `s2%` of course)
      - same
    - `Orbit`
    - `History_panels` (functions just like `pgstar` `history_panels`, only using `b%` info rather than `s%` info)



github issues!

we can't fix what we don't know...



happy binary-ing!!

the `binary` module

an introduction to binary evolution

Matthias FABRY, 23 July 2025