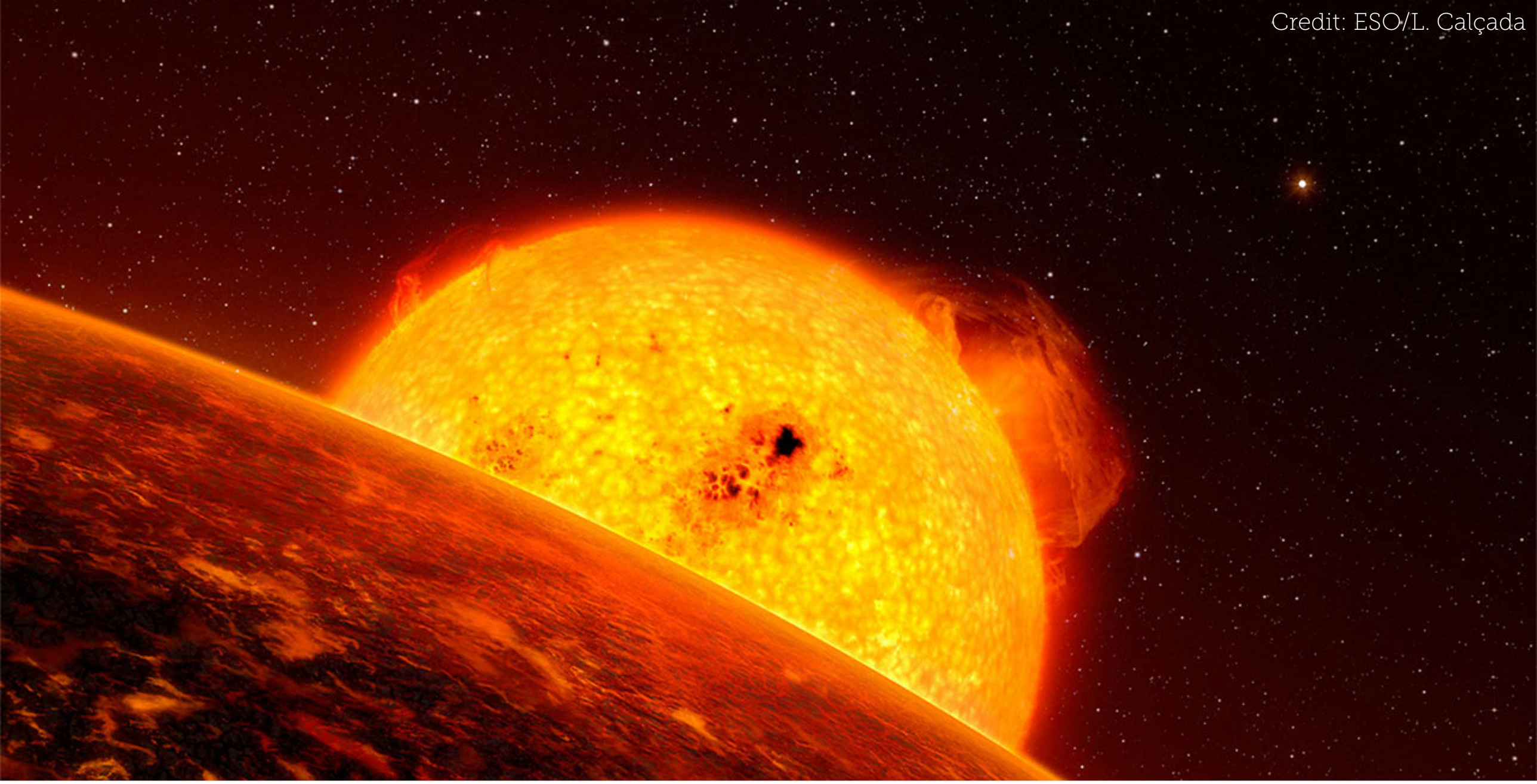


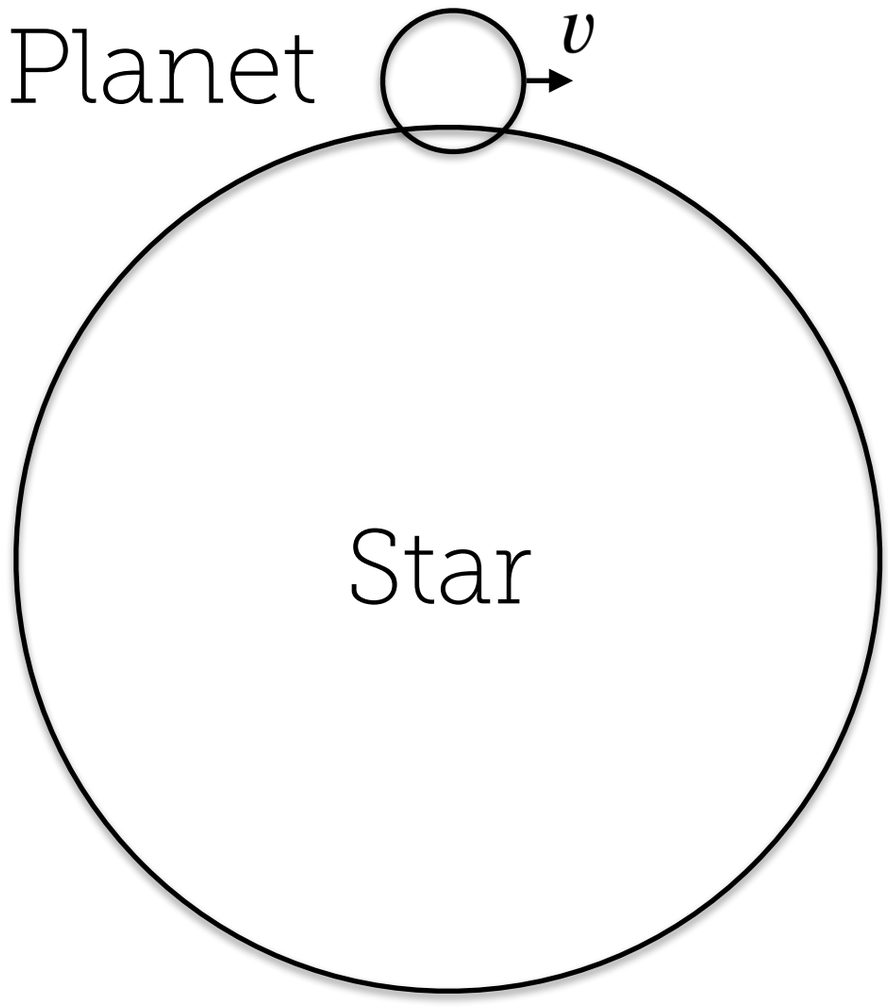
Planetary Engulfment with MESA Minilab 1

Matteo Cantiello [1,2]
Alina Istrate, Bill Wolf

[1] CCA, Flatiron Institute
[2] Princeton University

Through these labs we will try to build a simulation of a planetary engulfment using MESA.

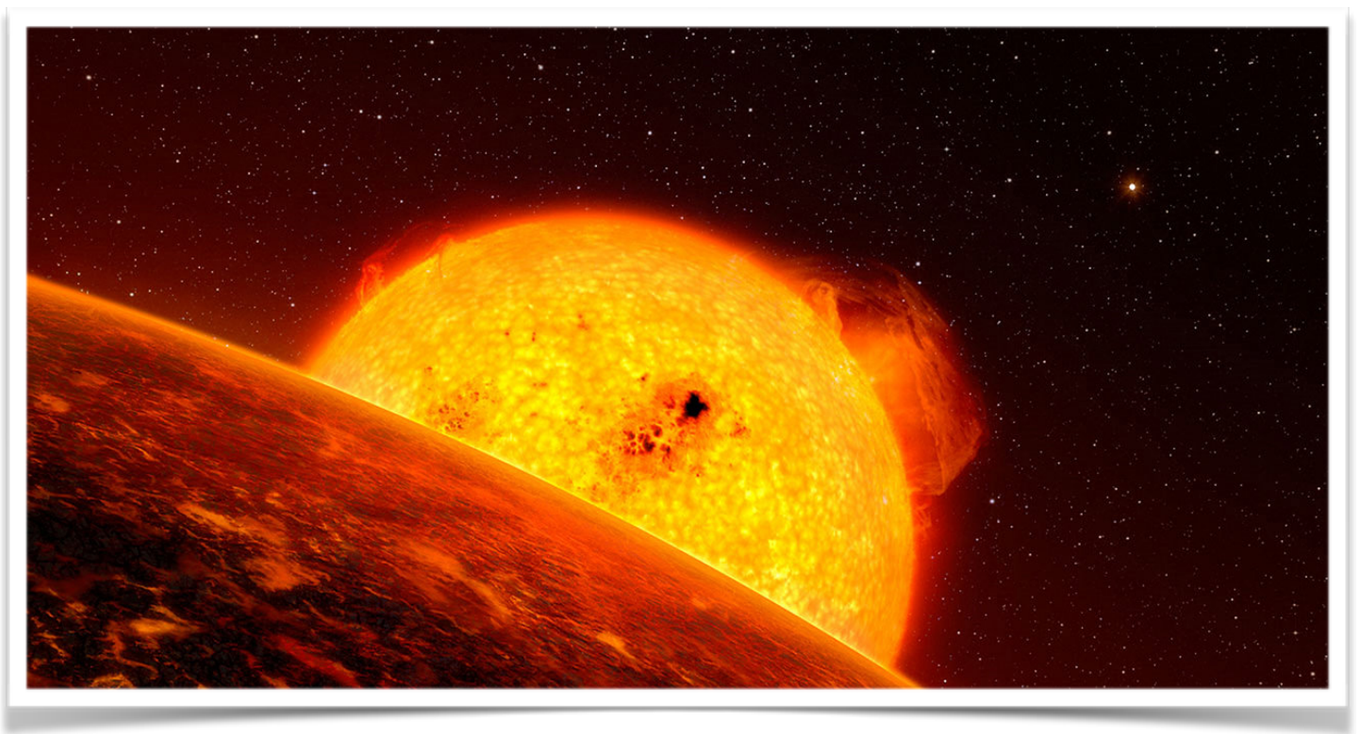
We will focus on a model built to simulate the ingestion of an Hot Jupiter by a red giant star, although the final results can probably be applied to a broader range of engulfment types.



Minilab 1 - Part A

Toward the RGB

- ✓ Download the labs from the 2018 MESA Summer school webpage. Open the Minilab1 folder.
- ✓ Create a new work directory and use the provided inlist, inlist_minilab1 and inlist_pgstar to evolve a $1M_{Sun}$ star to the red giant phase. Add a stopping criterion to halt the model when the radius of the star has reached 4 R_{Sun} and save your model (e.g. as `1msun_rg_4.mod`). While the model runs, move on to part B of the lab



Minilab 1 - Part B

Orbital Velocity

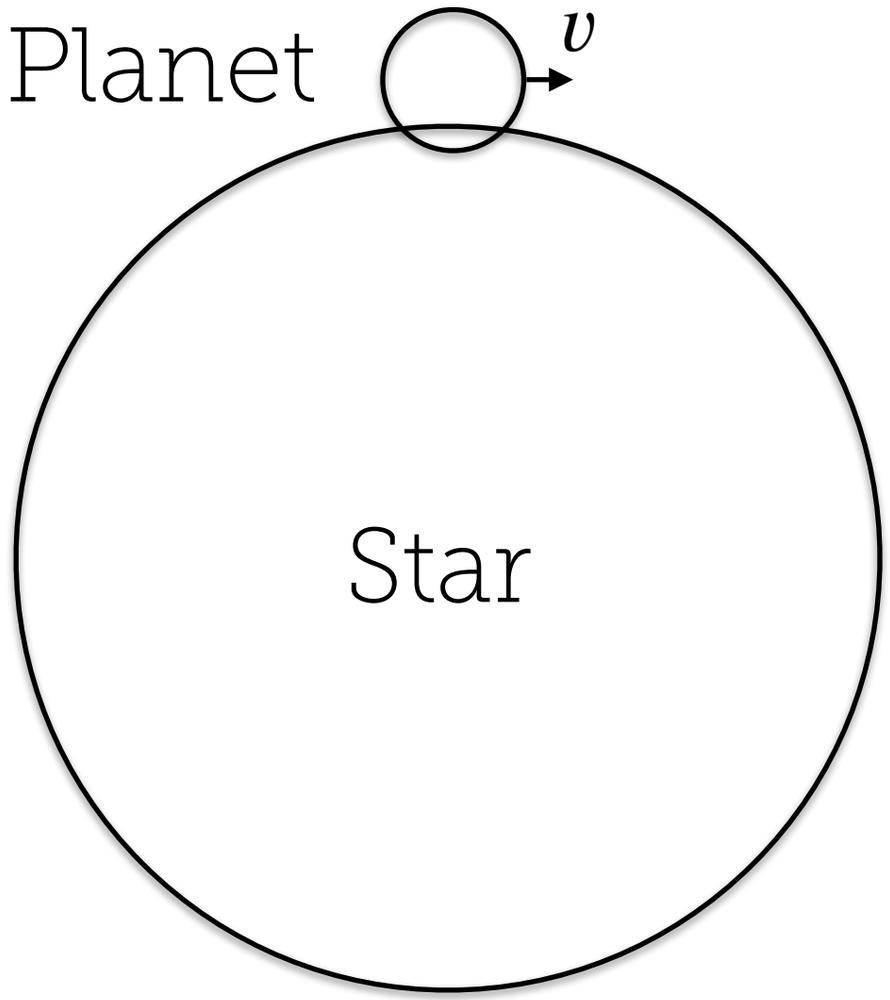
✓Use **`run_star_extras.f`** to write a function/subroutine that calculates the Keplerian orbital velocity of a planet orbiting a star of mass `m` at a distance `r`. Assume circular orbits.

Fortran Primer:
Subroutine vs Function
https://bit.ly/2KyZt3u

✓ Use your function/subroutine to calculate the velocity of a planet orbiting at r=R*. Include this output as '`Orbital_velocity`' in your history.data output

✓ Include the value of `Orbital_velocity` (in km/s) in your `pgstar` window

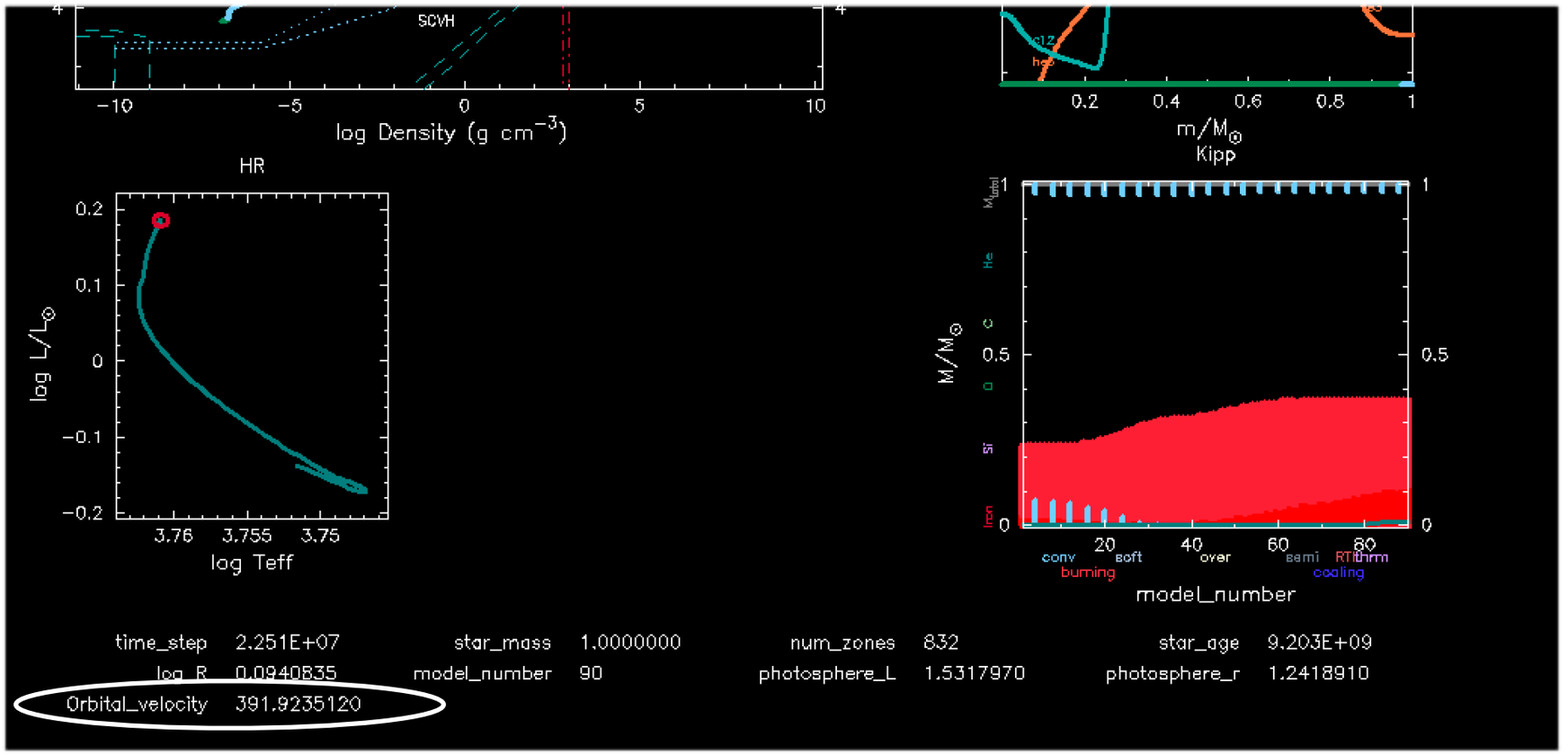
Solution



$$v = \sqrt{GM_1/r}$$

```
subroutine orbital_velocity(m1, r, v_kepler)
           real(dp) ::  m1, v_kepler,  r
           use const_def, only: standard_cgrav
           v_kepler = sqrt(standard_cgrav*m1/r)
end subroutine orbital_velocity
```

Solution



In run_star_extras.f -> how_many_extra_history_columns add:

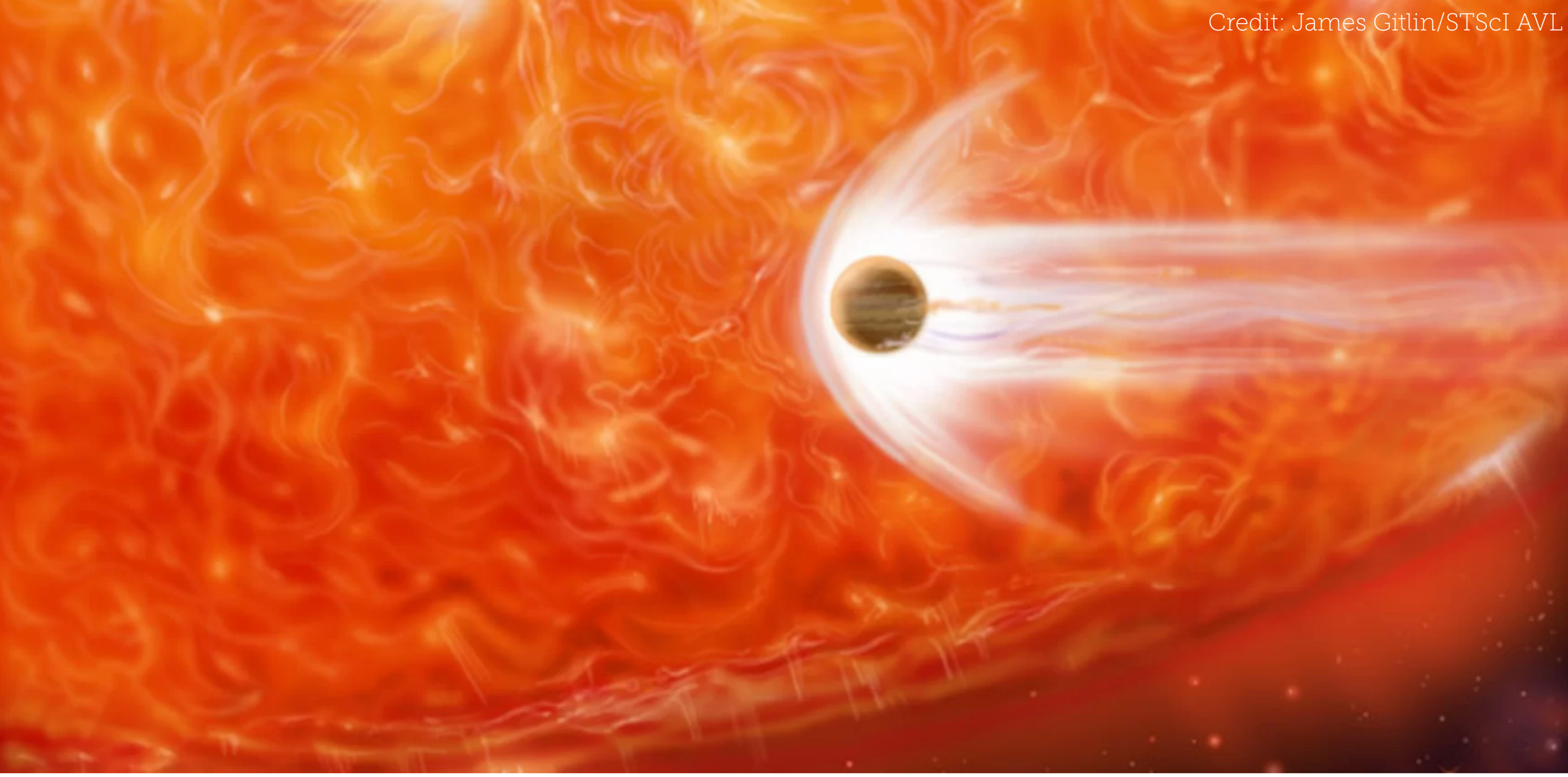
```
how_many_extra_history_columns = 1
```

In run_star_extras.f -> data_for_extra_history_columns add:

```
names(1) = 'Orbital_velocity' ! v_kepler
call orbital_velocity(s% m(1), s% r(1), vals(1))
vals(1) = vals(1)/1d5 ! Make it km/s
```

In inlist_pgstar_minilab1 add:

```
Text_Summary1_name(3,1) = 'Orbital_velocity'
```



Planetary Engulfment with MESA Minilab 2

Matteo Cantiello [1,2]
Alina Istrate, Bill Wolf

[1] CCA, Flatiron Institute
[2] Princeton University

FLATIRON INSTITUTE
Division of Simons Foundation

Minilab 2

In MiniLab 1 we looked at the evolution of a 1 Solar Mass star toward the red giant phase. We also learned how to keep track of a property (Keplerian velocity) of a putative planet orbiting such star using

`run_star_extras.f`

Since today our final goal is to use MESA to calculate the outcome of planetary engulfments, we need to know how to include the heating coming from e.g. the aerodynamic drag of a planet plunging into its host' envelope.

Therefore, in Minilab 2 we will learn the basics of how to incorporate energy heating into our stellar model using the **`other_energy`** module.

Minilab 2 — Adding an other_energy routine

- ✓ To create a custom **`other_energy`** routine, let's start by copying and pasting the **`default_other_energy`** subroutine into your **`run_star_extras.f`** file. The default version lives in `MESA_DIR/star/other/other_energy.f90`. You only need to copy the subroutine, not the entire contents of other_energy.f90
- ✓ Rename the routine to e.g. **`engulfment_energy`**
- ✓ In **`run_star_extras.f`**, the subroutine **`extras_controls`** tells MESA which modifications to include. Add a line that tells MESA to use a new subroutine for `other_energy` named **`engulfment_energy`**
- ✓ In order for MESA to use this routine you must also add the statement **`use_other_energy = .true.`** to the control section of your inlist
- ✓ Add a simple print statement in your otherwise empty `my_energy` routine, compile your local MESA work directory and check that the routine is called at runtime

Minilab 2

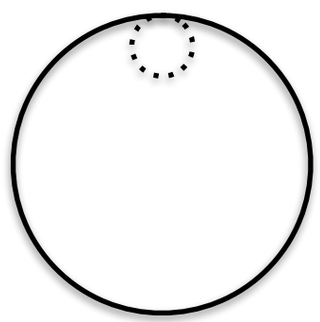
Calculating heating

MESA accounts for extra energy deposition using the array **`s% extra_heat`**, which is the extra energy added per unit mass per unit time in each cell. Its value is zero unless you change it in your `other_energy` routine. `s% extra_heat` has units of erg/g/s. The heat added to each cell k after each time step is then:

`dE(k) = s% extra_heat(k)*s% dm(k)*s% dt`,

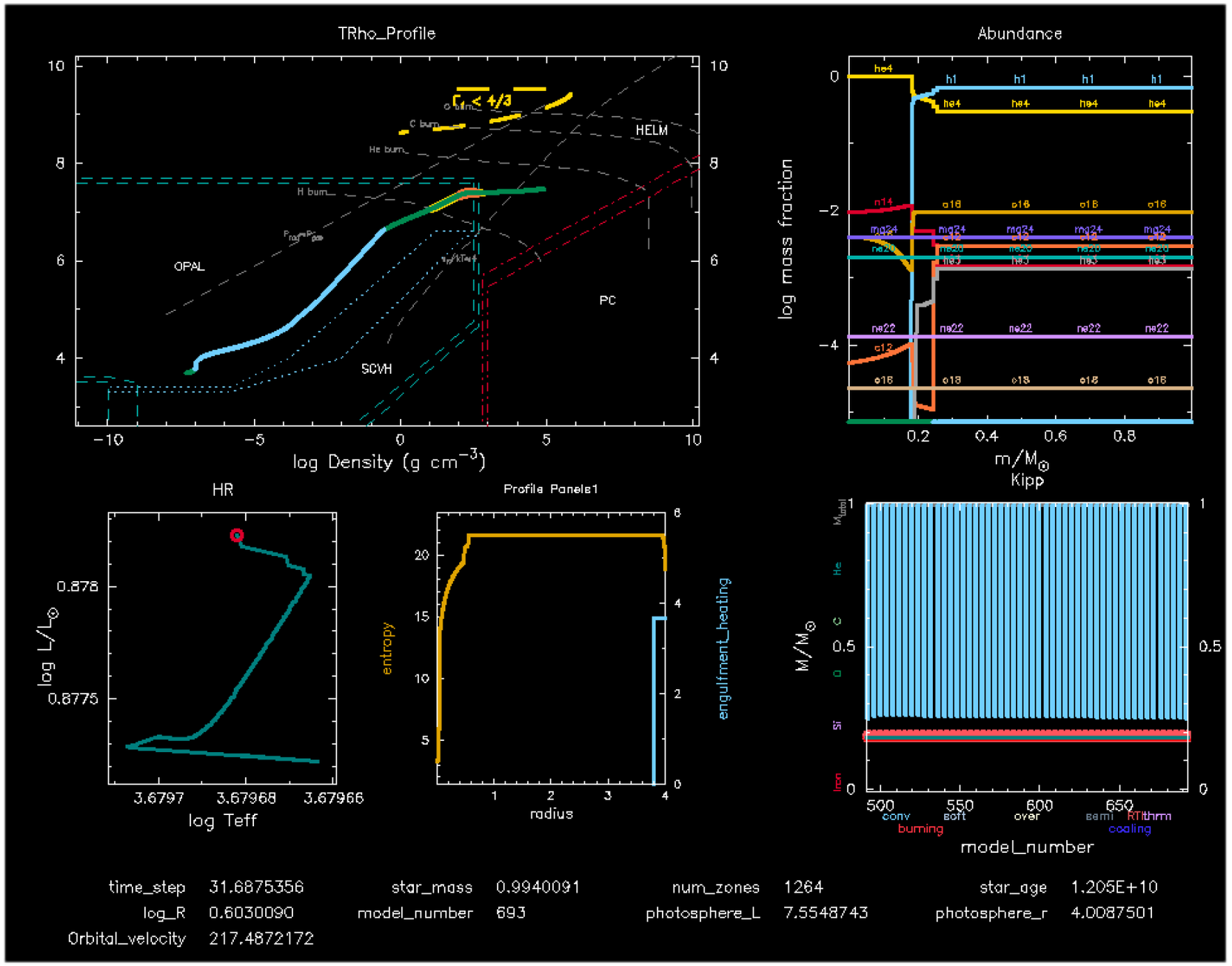
where `s% dm(k)` is the mass of the cell, and `s% dt` is the time step.

- ✓ Assume Jupiter (0.001 M_{Sun}, 0.1R_{Sun}) is engulfed right below the surface of its evolved host star with radius `s% r(1)`. Use your saved model with R=4R_{Sun}. If not yet ready, you can find a pre made one in the Minilab2 folder. Use `s% extra_heat` to dissipate, at each time step, 1% of the orbital energy of the planet-star system in the stellar shell corresponding to the radial location of the planet.



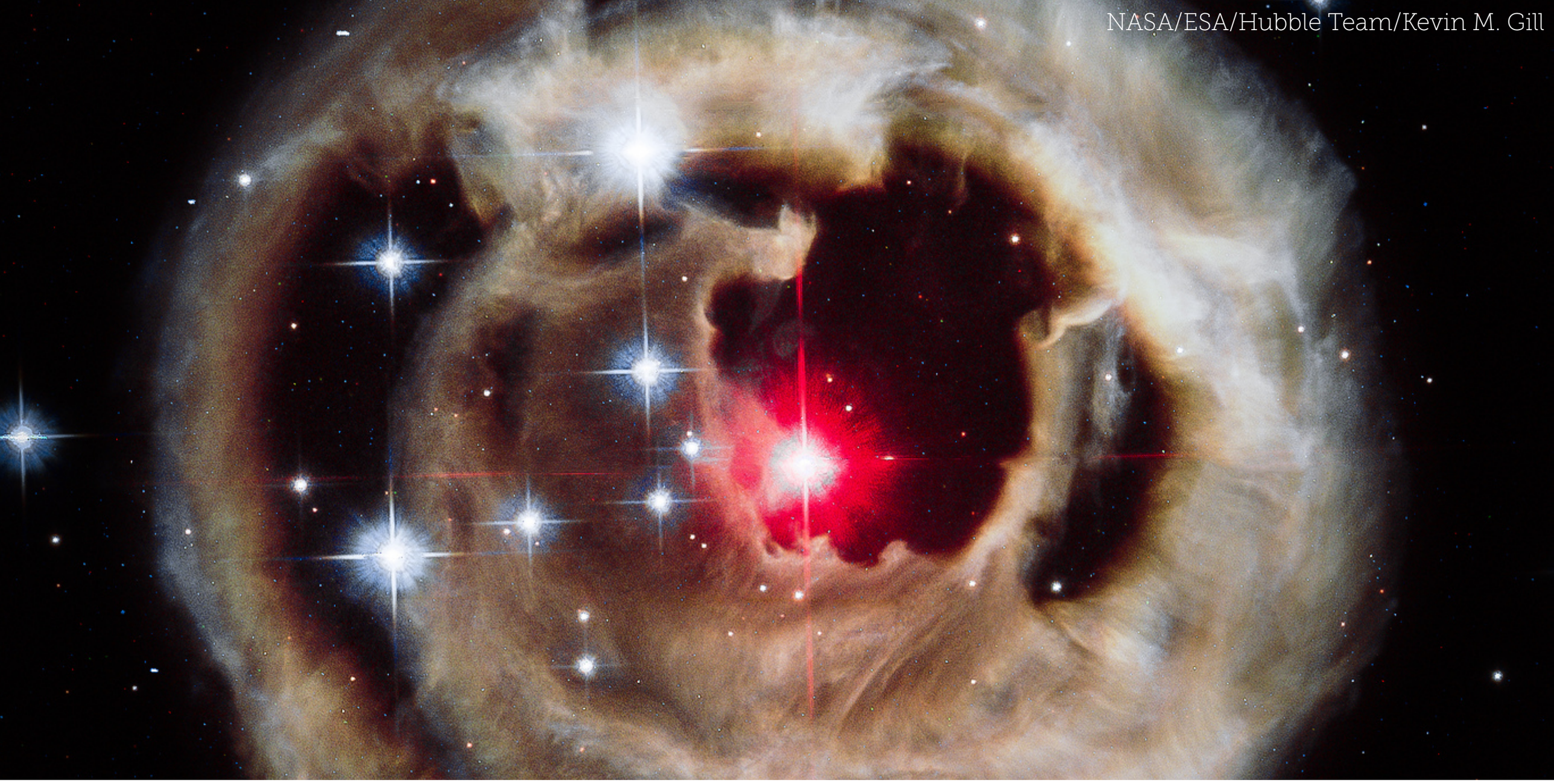
- ✓ Limit the timestep to `max_timestep = 1d9`
- ✓ Add the engulfment heating to your pgstar plot (e.g. as `safe_log10_cr(s% extra_heat(k))`) and run the calculation for a few time steps

Solution



➡ See Minilab2-Solutions -> run_star_extras.f & inlists

```
subroutine orbital_energy(m1, m2, r, energy)
  real(dp) ::  m1, m2,  r, energy
  energy = -standard_cgrav*m1*m2/(2d0*r)
end subroutine orbital_energy
```



NASA/ESA/Hubble Team/Kevin M. Gill

Planetary Engulfment with MESA Maxilab

Matteo Cantiello [1,2]
Alina Istrate, Bill Wolf

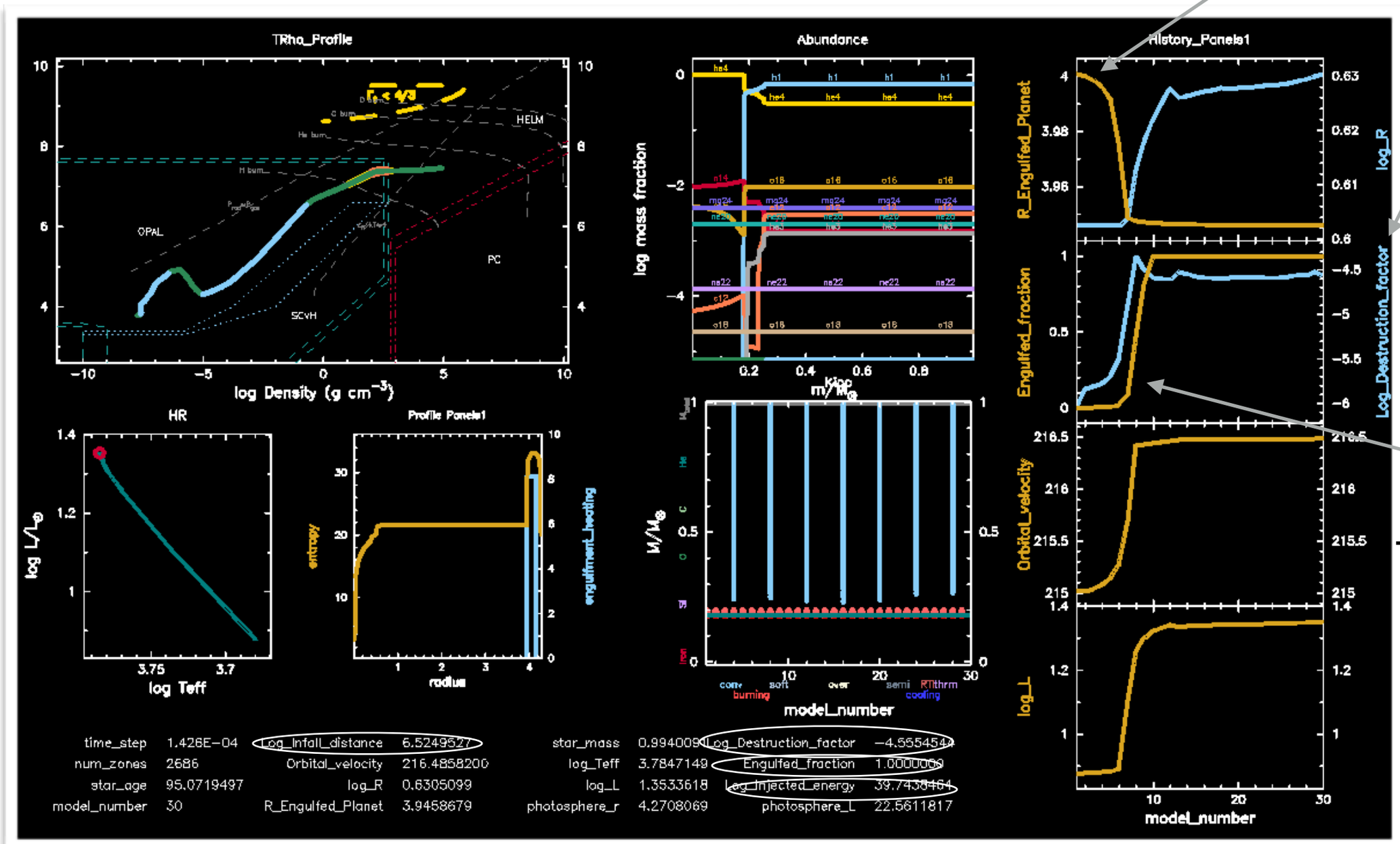
[1] CCA, Flatiron Institute
[2] Princeton University

Maxilab - Part A

- ✓ Read the provided **`run_star_extras.f`** pay particular attention to the comments
- ✓ Use the provided inlists and run the calculation
- ✓ Carefully study pgstar output
- ✓ What is happening? What is the problem / why this is not properly modeling a planetary engulfment?
- ✓ Implement the change that can improve the calculation. Pay particular attention to where you need to make this change. The MESA flow chart can help...

Navigating the Pgstar Output

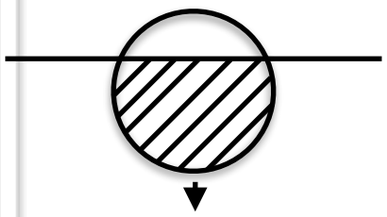
Radial location of the planet



Disruption Factor

$$f \equiv \frac{\rho_{\rm ext} v^2}{\bar{\rho}_{\rm p} v_{\rm esc}^2}$$

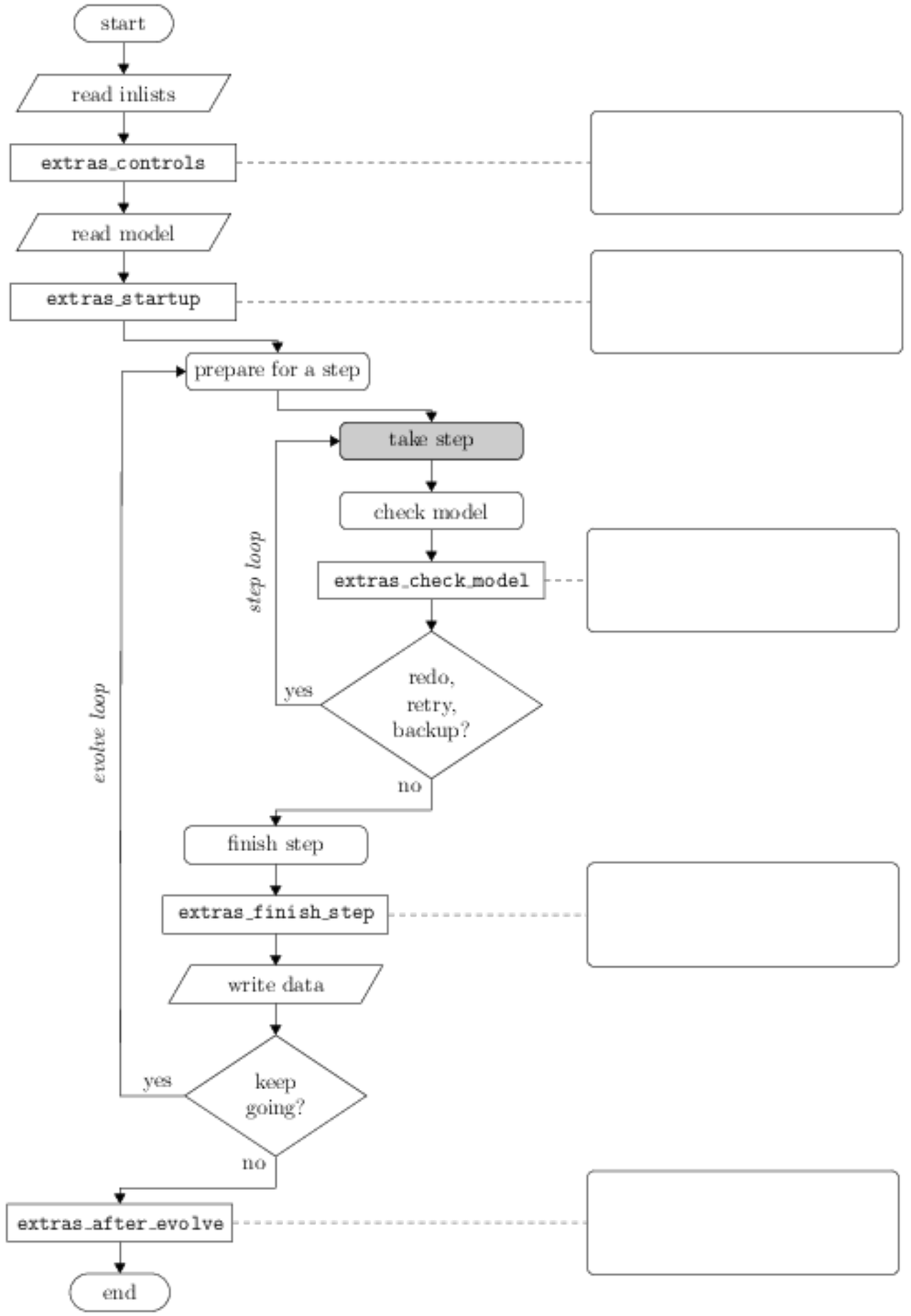
Engulfed Fraction



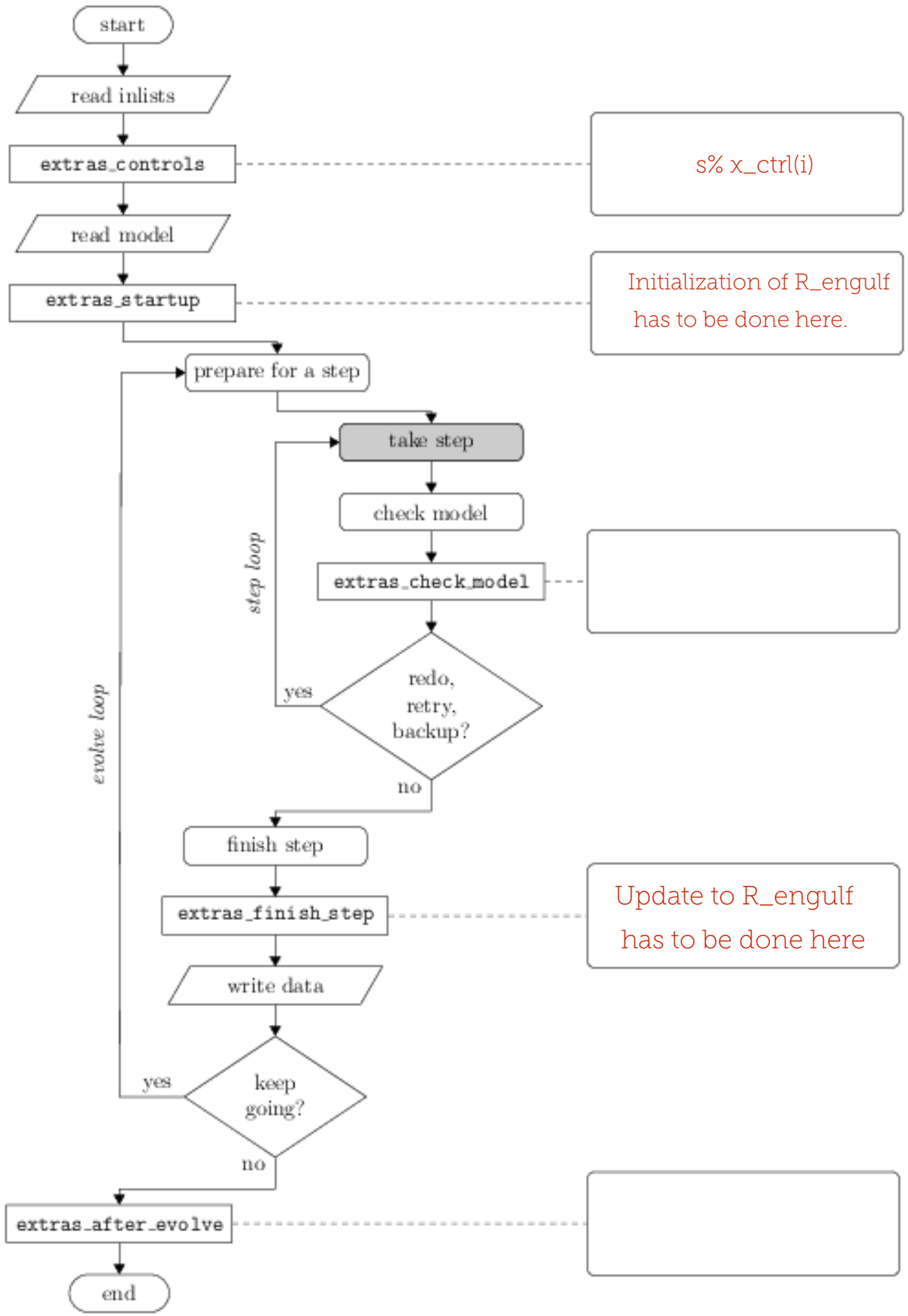
log(dr/cm)

log(de/erg)

Maxilab
MESA Flow Chart

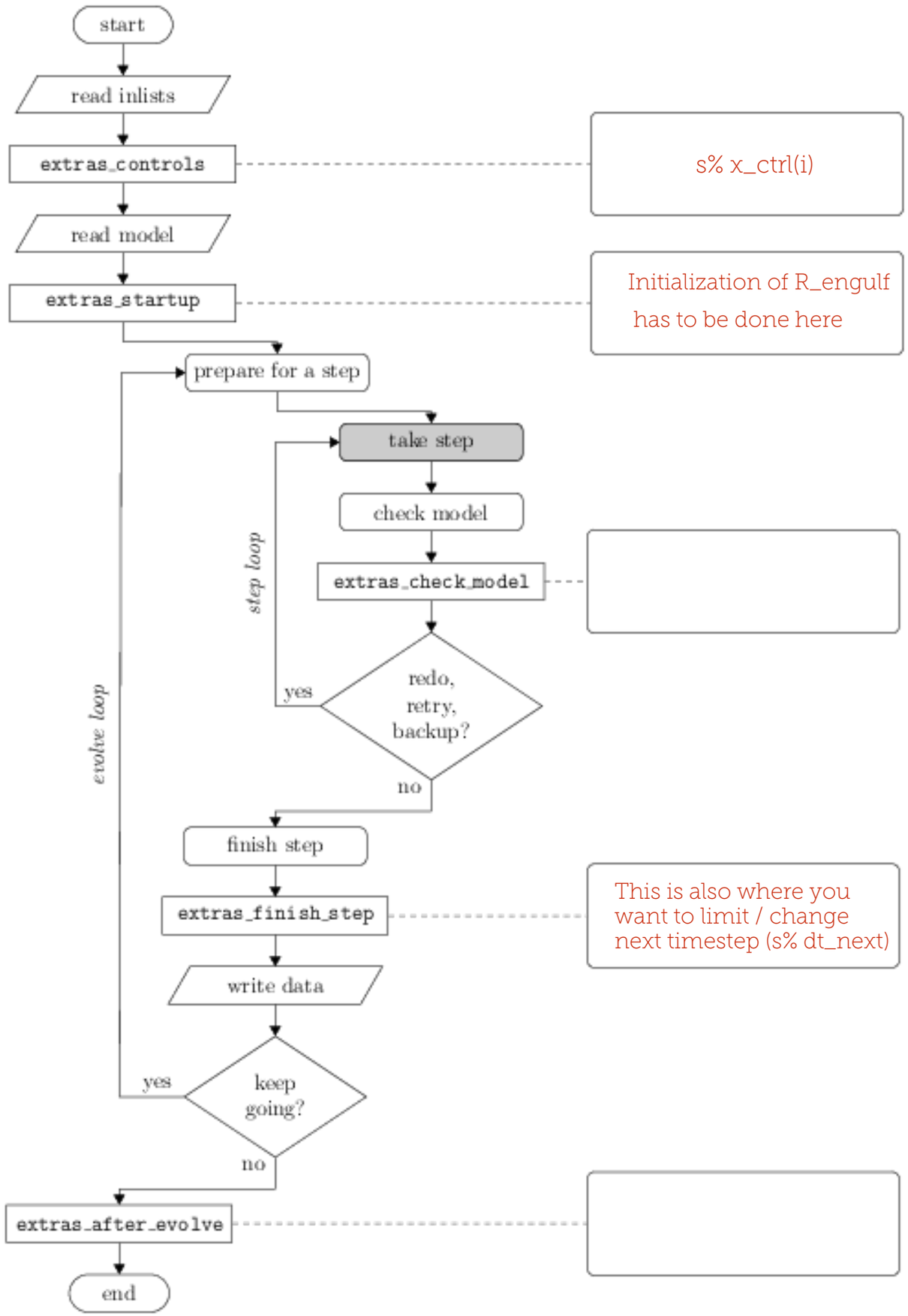


Solution



- Need to update the radial location (R_engulf) of the planet in the star. The issue is that MESA needs to remember previous value of R_engulf, and this value has to be updated to (R_engulf - dr) only if the model converged.
- Do the update in **`extras_finish_step`**
- Use **`move_extra_info update`** to save quantities and be able to restart from photos

Solution



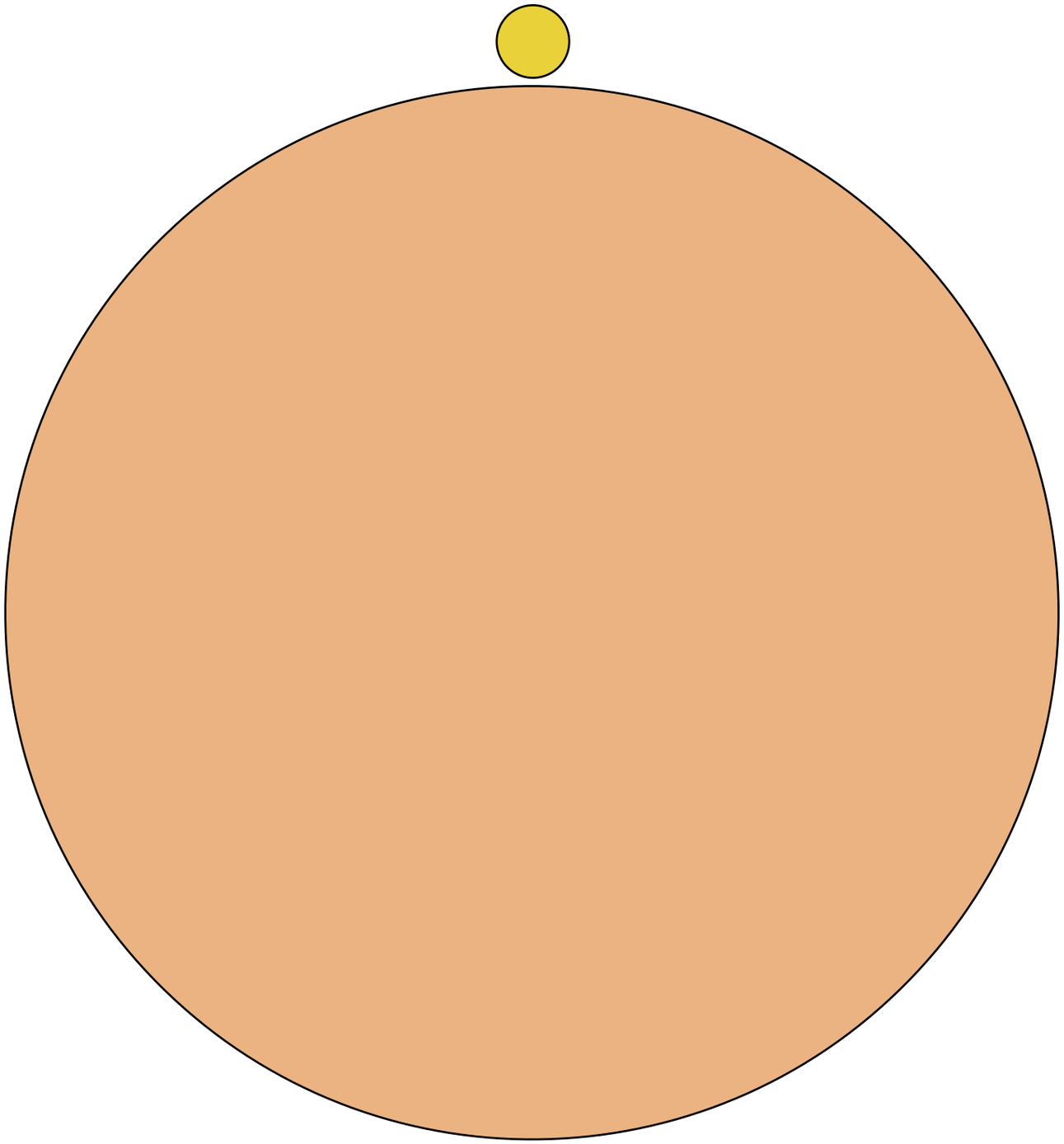
- Need to update the radial location (R_engulf) of the planet in the star. The issue is that MESA needs to remember previous value of R_engulf, and this value has to be updated to (R_engulf - dr) only if the model converged.
- Do the update in **`extras_finish_step`**
- Use **`move_extra_info update`** to save quantities and be able to restart from photos

Maxilab - Part B

- ✓ Check your changes against the **`run_star_extras.f`** provided in the Maxilab solution. Use the provided inlists and run the calculation again
- ✓ Carefully study pgstar output
- ✓ Bonus Assignment 1: Implement a way to stop the calculation after the planet is disrupted but giving the star the time to thermally relax to the heat injection
- ✓ Bonus Assignment 2: Try running the engulfment using stars with different radii (in Models folder) and/or changing the radius/mass of the engulfed planet

Extra info about the code in run_star_extras.f

(Not part of the assignment, but useful to understand the calculations)

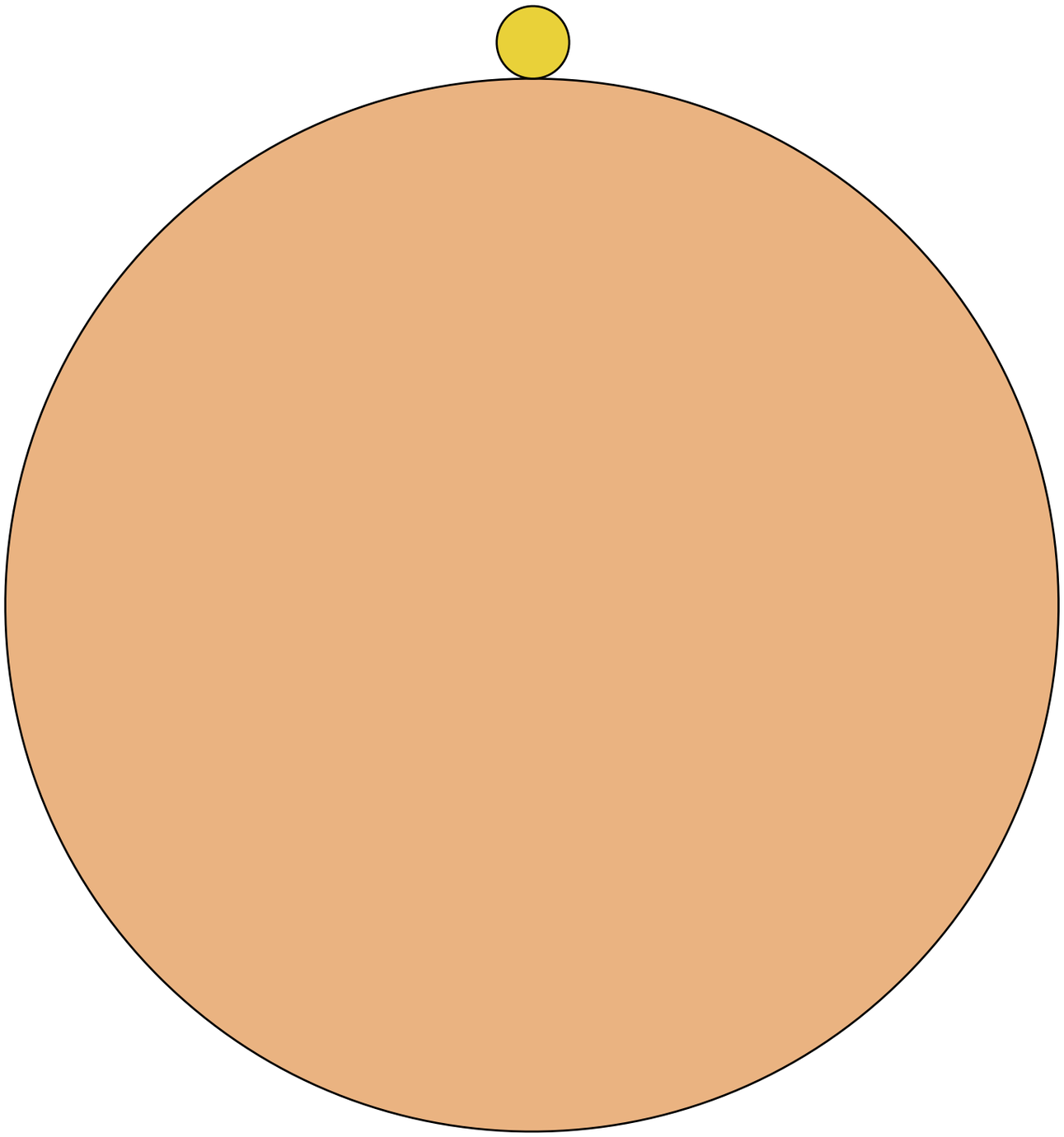


$$E = -\frac{G\,M_1\,M_2}{2\,r}$$

$$v = \sqrt{GM_1/r}$$

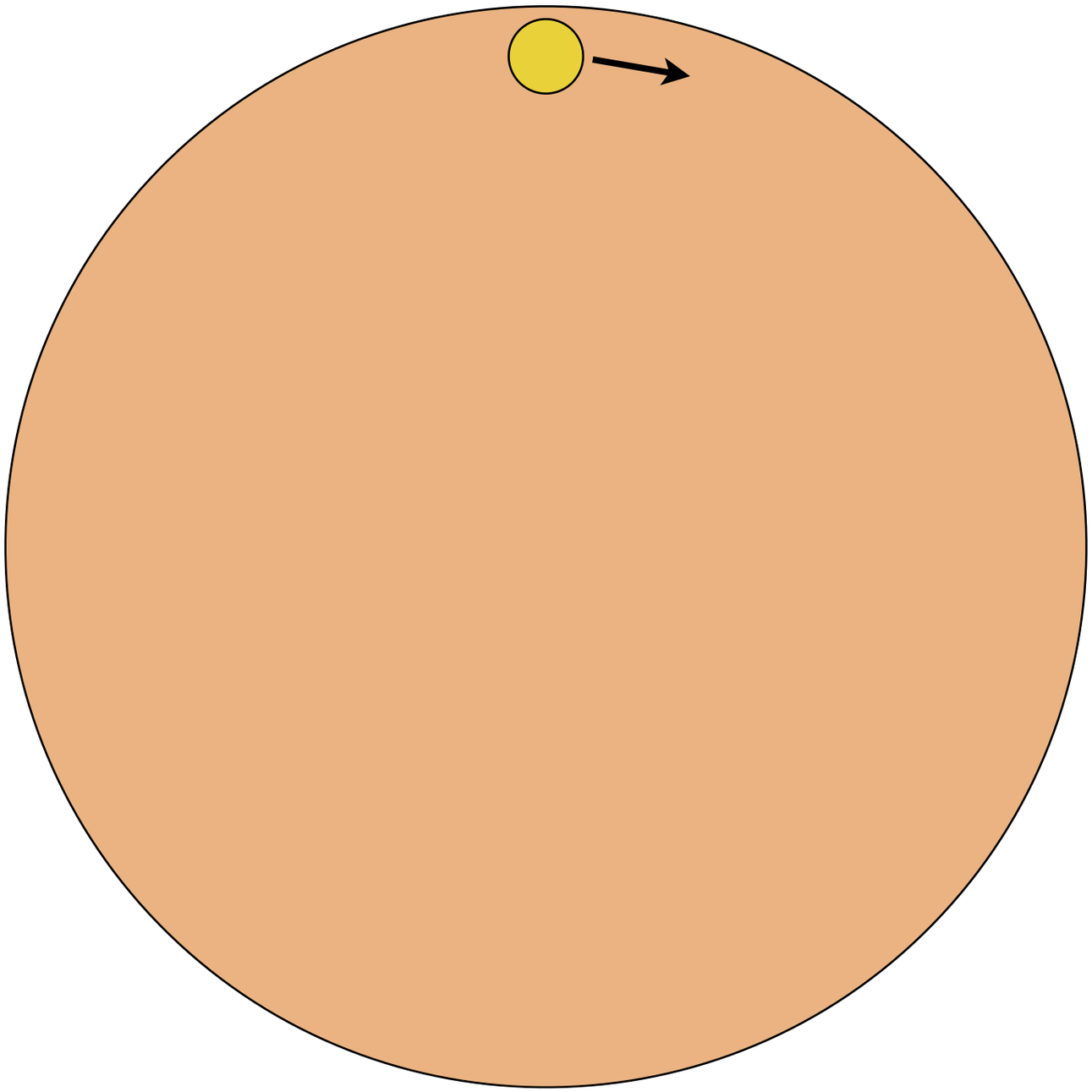
We follow

Tylenda & Soker 2006



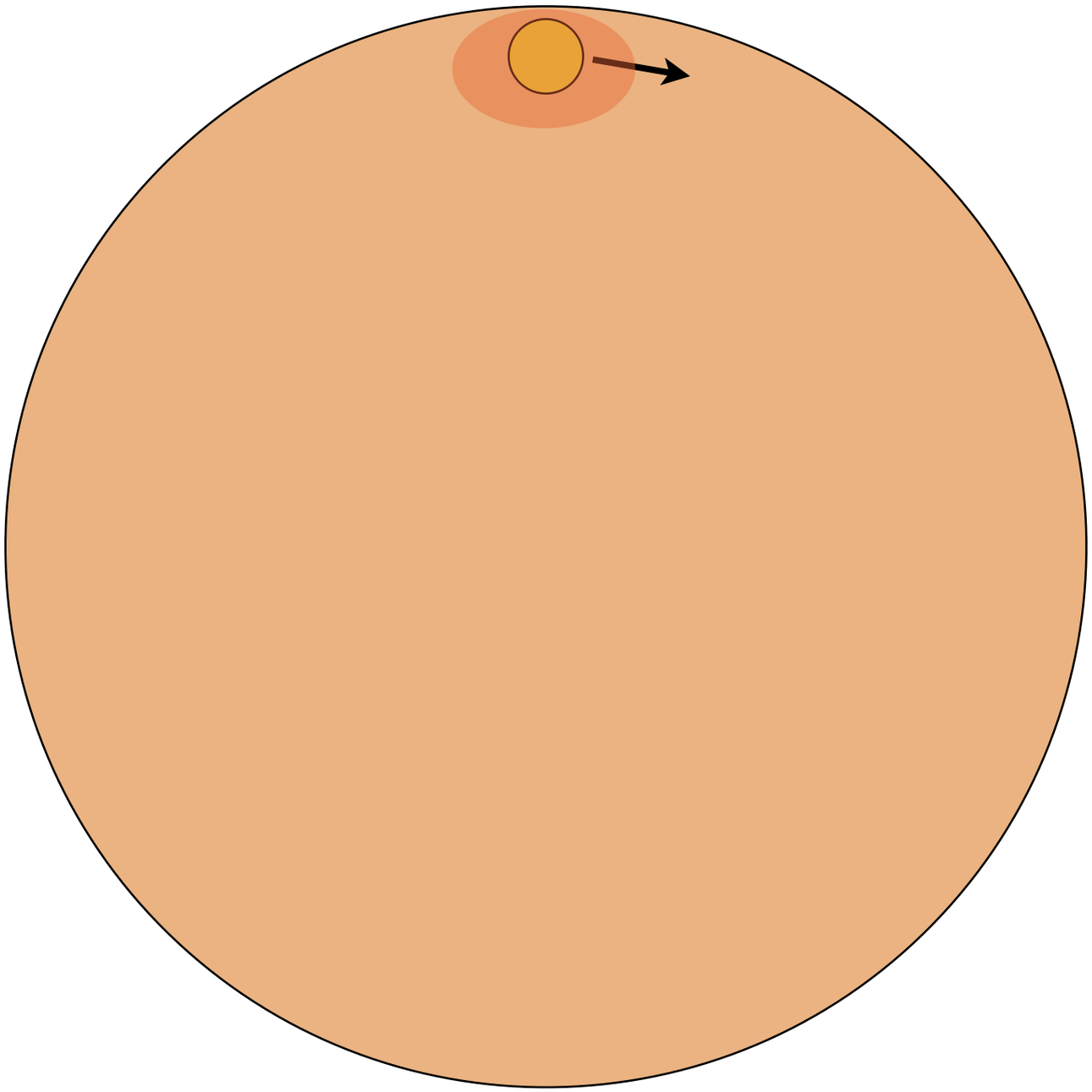
we start the engulfment with a grazing collision

Drag



$$\frac{\mathrm{d}E}{\mathrm{d}t} \simeq -\pi R_2^2 \rho v \frac{v^2}{2}$$

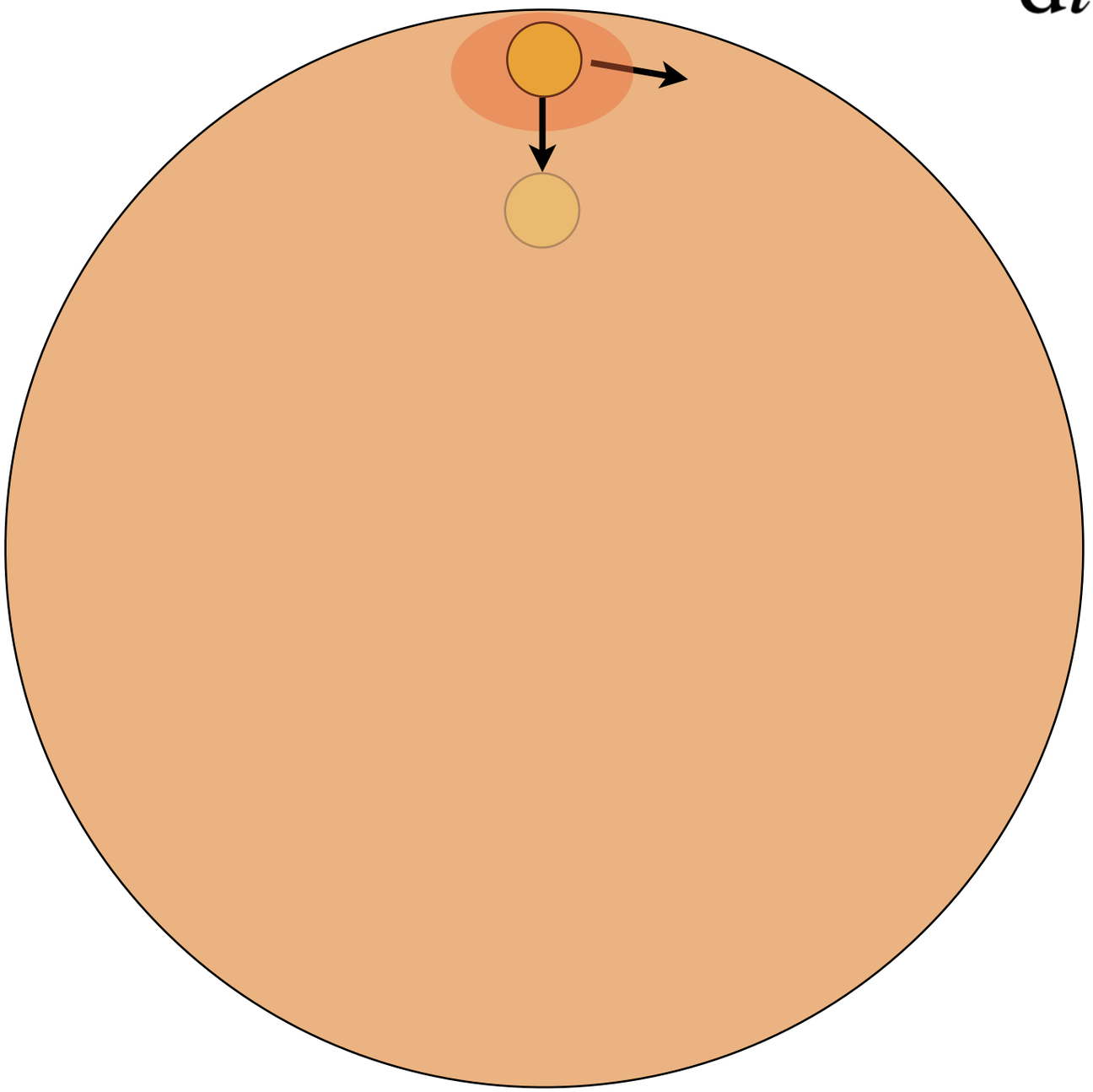
Heating



$$\frac{\mathrm{d}E}{\mathrm{d}t} \simeq -\pi R_2^2 \rho v \frac{v^2}{2}$$

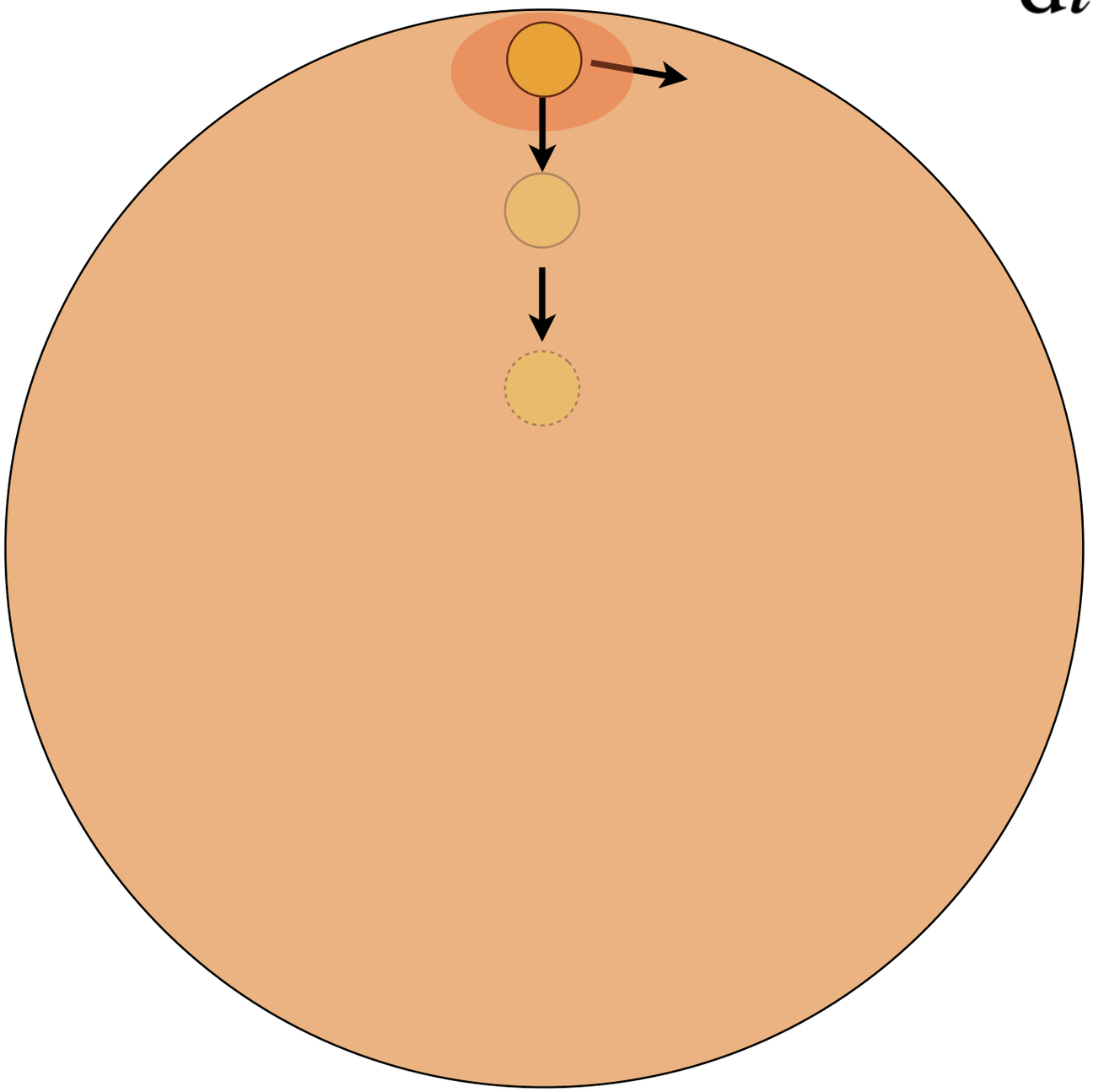
Orbital Decay

$$\frac{\mathrm{d}r}{\mathrm{d}t} = -\frac{\pi R_2^2}{M_2} \rho \left(G M_1 r\right)^{1/2}$$



Orbital Decay

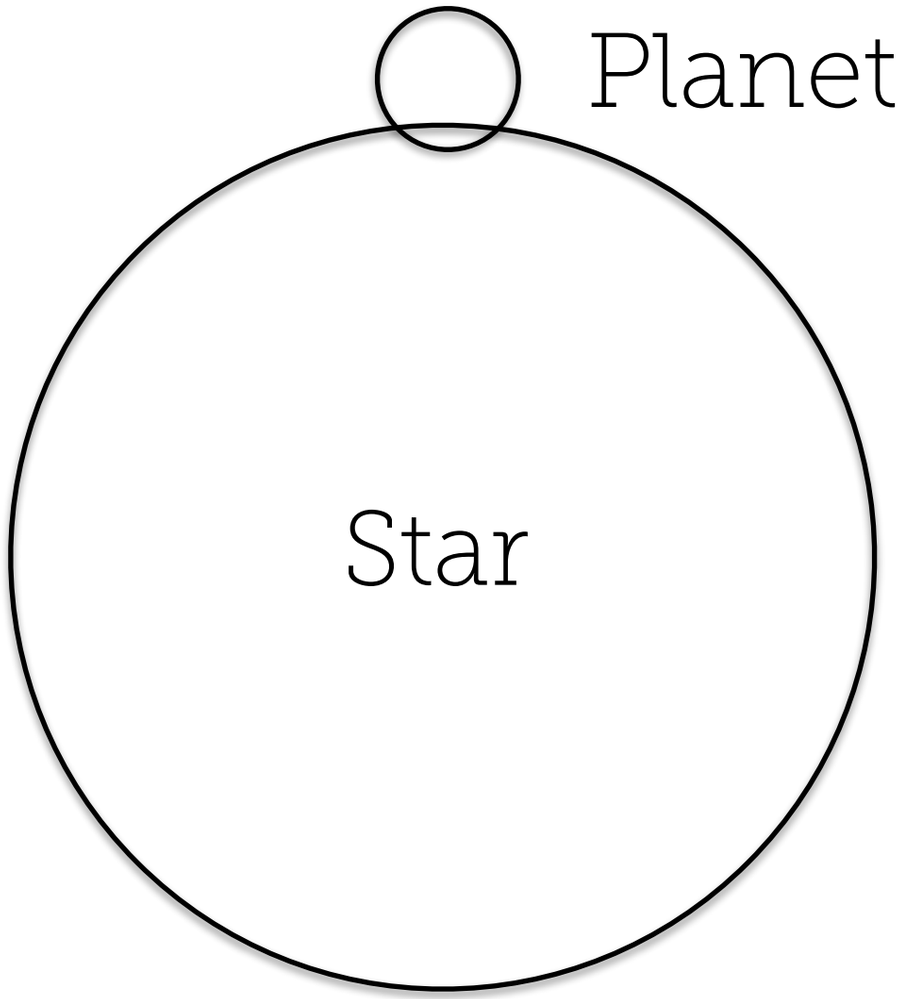
$$\frac{\mathrm{d}r}{\mathrm{d}t} = -\frac{\pi R_2^2}{M_2} \rho (G M_1 r)^{1/2}$$



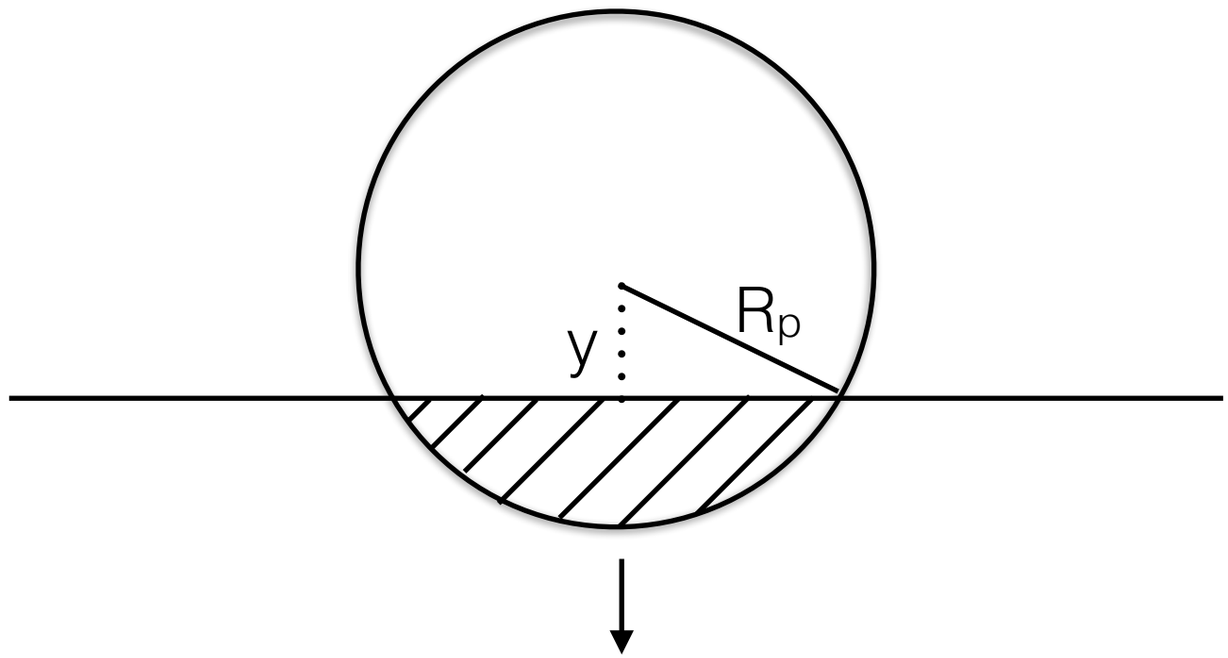
Stop e.g. when planetary destruction f=1

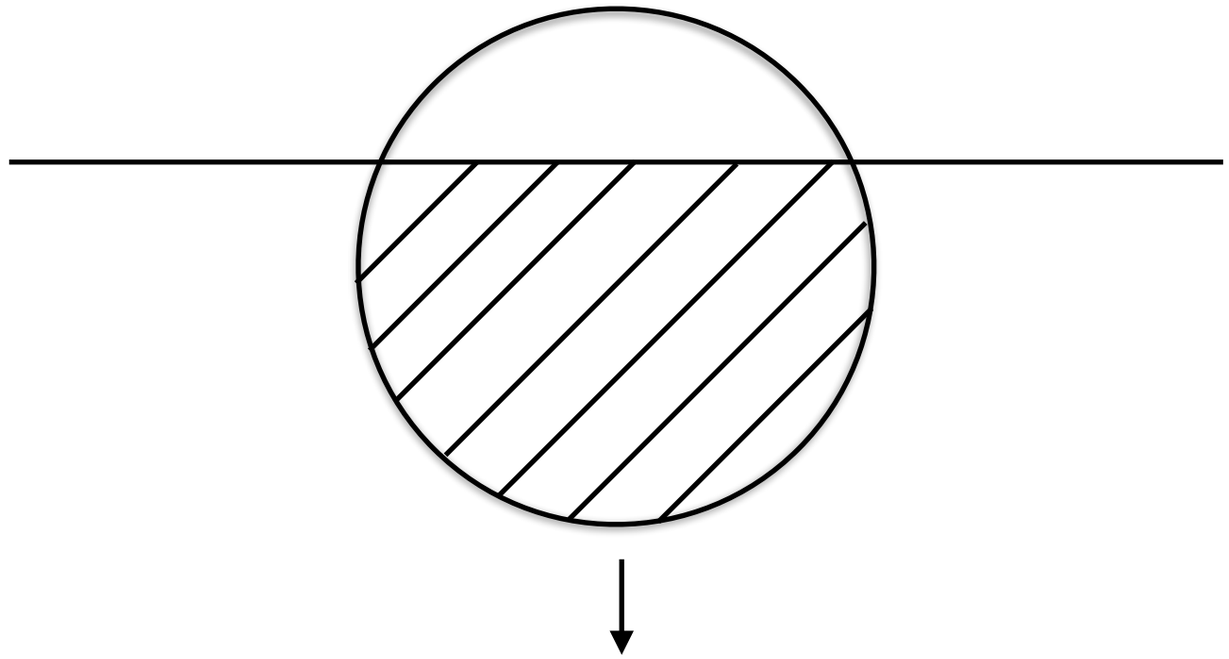
(See e.g. Jia & Spruit 2018)

Grazing Phase



for $R_* >> R_p$





Some details of the engulfment calculation

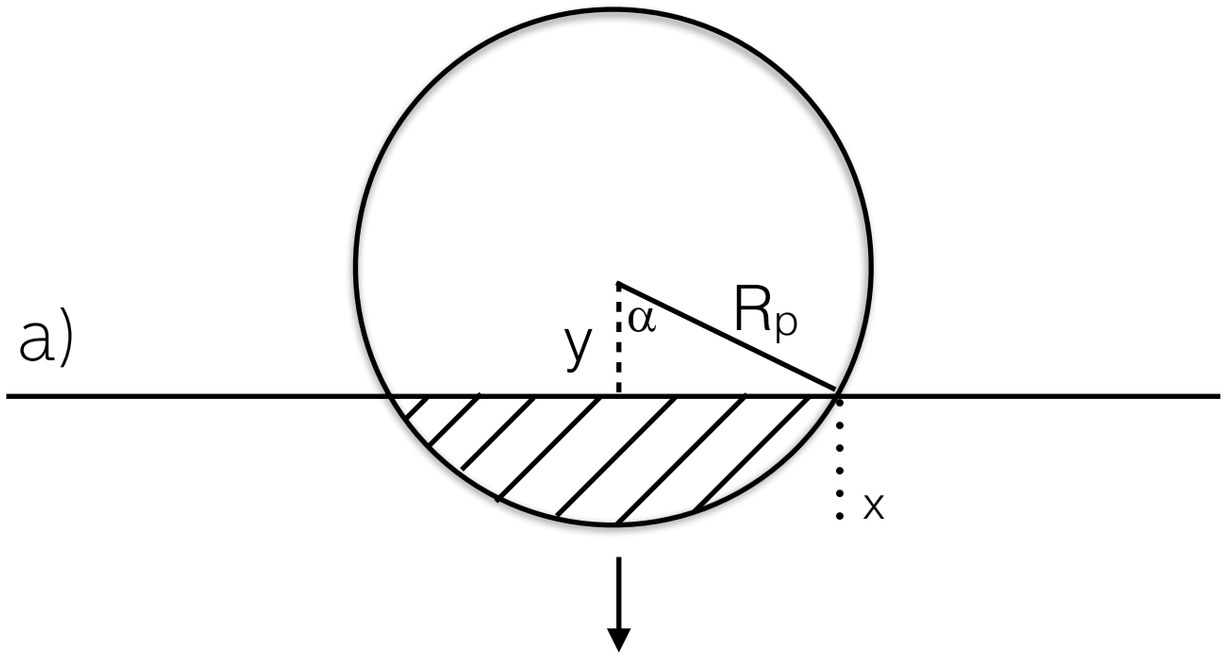
Some details of the engulfment calculation

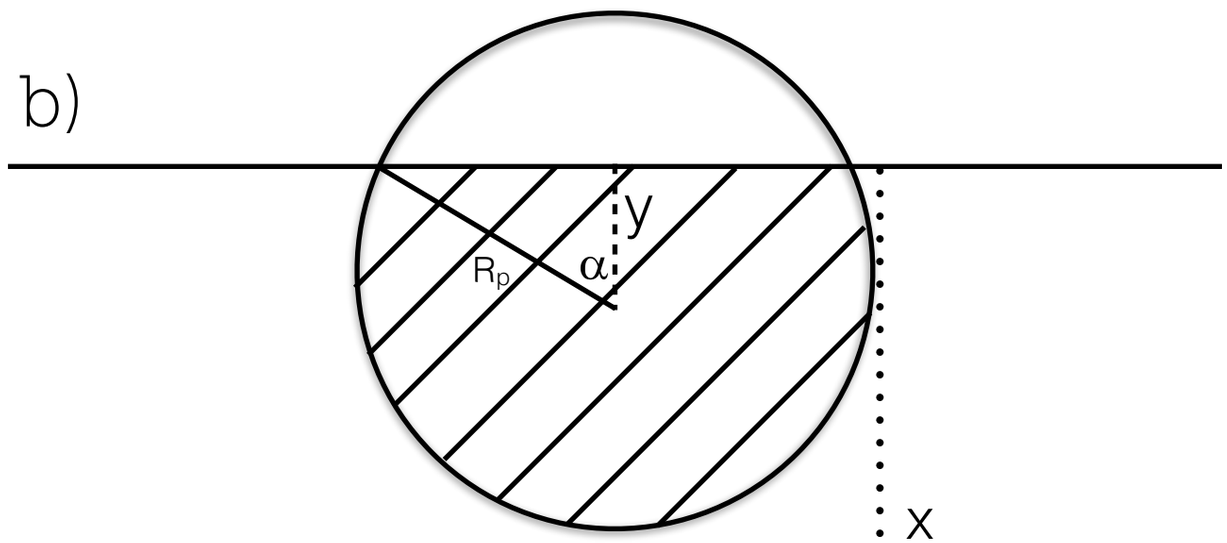
- ✓ We identify the envelope region occupied by the planet at the beginning of the engulfment. We assume it is a shell
- ✓ We determine mass in this shell and spread drag energy uniformly in this region using **`s% extra_heat`**
- ✓ We limit MESA time step **`s% dt`** so that the change of radial coordinate of the planet **`dr`** does not exceed 0.05 R_{planet} on the grazing phase and 1 R_{planet} during full engulfment (kinda arbitrary values allowing the code to run fast enough. A small convergence study shows this is likely not too bad in most cases)

Some routines/functions explained

Grazing Phase Area Calculation

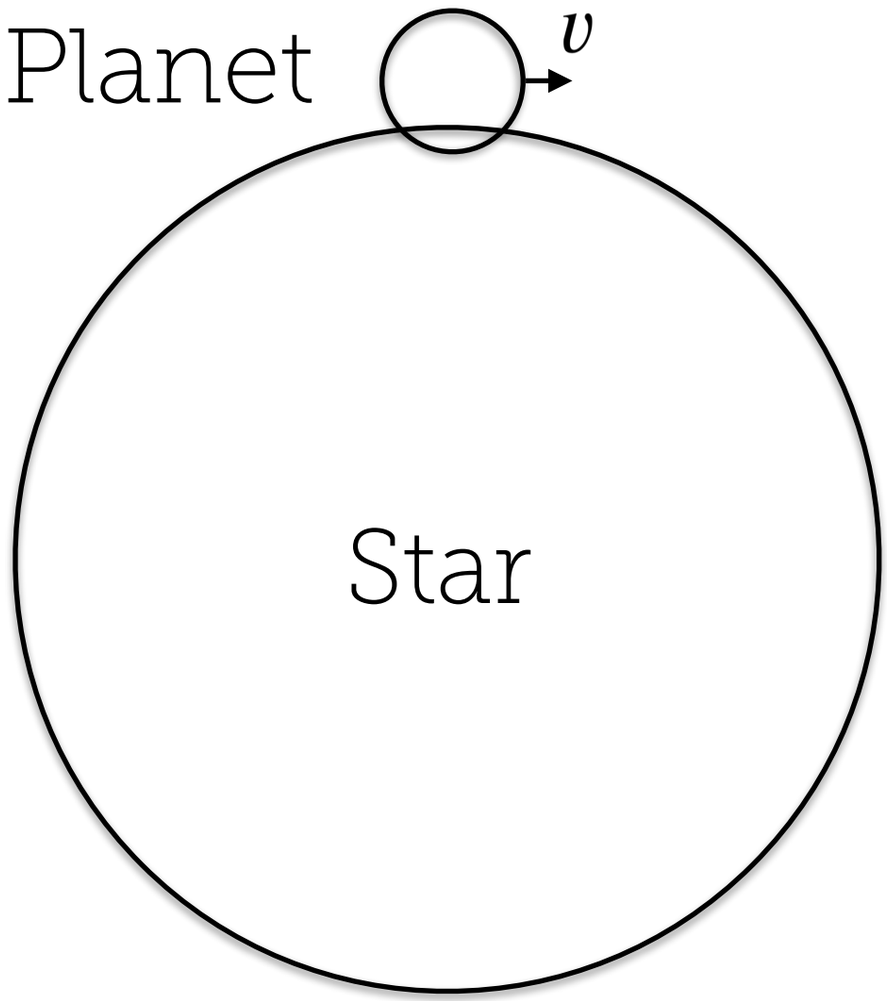
for $R_* >> R_p$






```
real(dp) function intercepted_area(x, r_planet) result(area)
            real(dp), intent(in) :: x, r_planet
            real(dp) :: alpha, y

            if (x < r_planet) then     ! Case A (less than half of the planet engulfed)
              y = r_planet - x
              alpha = acos (y/r_planet)
              area = r_planet * (r_planet*alpha - y * sin(alpha))
            else
              y = x - r_planet         ! Case B (more than half of the planet engulfed)
              alpha = acos (y/r_planet)
              area = pi*r_planet**2d0 - r_planet * (r_planet*alpha - (y * sin(alpha)))
            endif
end function intercepted_area
```



Orbital Parameters

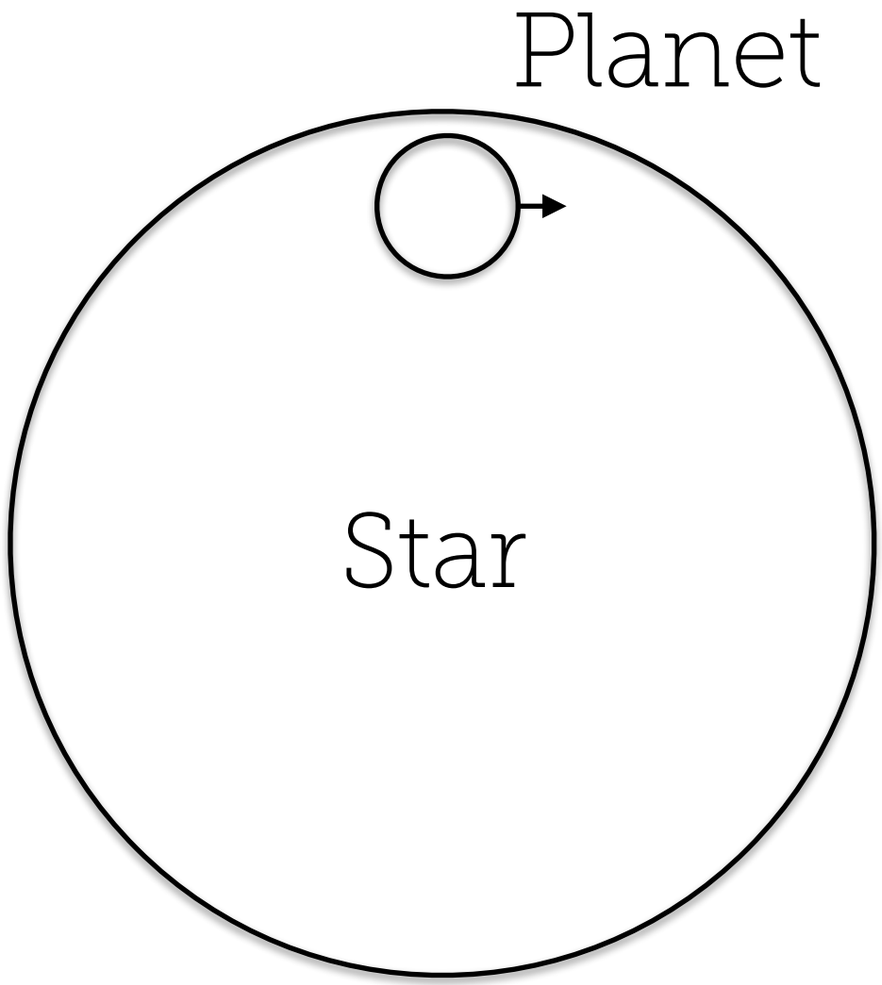
1) $$v = \sqrt{GM_1/r}$$

2) $$E = -\frac{G\,M_1\,M_2}{2\,r}$$

```
subroutine orbital_velocity(m1, r, v_kepler)
           real(dp), intent(in) ::  m1, r
           real(dp), intent(out) :: v_kepler
           use const_def, only: standard_cgrav
           v_kepler = sqrt(standard_cgrav*m1/r)
end subroutine orbital_velocity

subroutine orbital_energy(m1, m2, r, energy)
           real(dp), intent(in) ::  m1, m2,  r
           real(dp), intent(out):: energy
           use const_def, only: standard_cgrav
           energy = -standard_cgrav*m1*m2/(2d0*r)
end subroutine orbital_energy
```

Fortran Primer
Subroutine vs Function
https://bit.ly/2KyZt3u



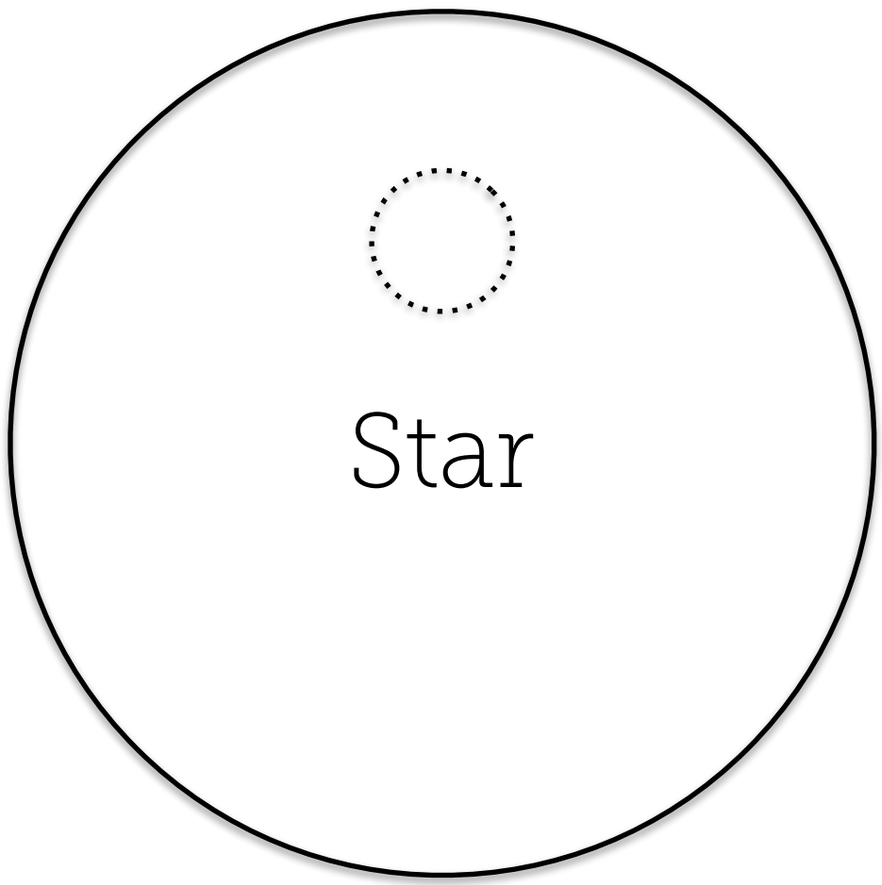
Aerodynamic Drag

$$\frac{\mathrm{d}E}{\mathrm{d}t} \simeq -\pi R_2^2 \rho v \frac{v^2}{2}$$

Combining with time derivative of Eq.2 we get:

$$\frac{\mathrm{d}r}{\mathrm{d}t} = -\frac{\pi R_2^2}{M_2} \rho (G M_1 r)^{1/2}$$

```
subroutine what_a_drag(m1, m2, area, rho, dt, r, de, dr)
! Calculate Change in radial position and energy loss due to drag (CGS)
            real(dp) :: cdr, cde
            real(dp), intent(in) :: m1, m2, area, rho, dt, r
            real(dp), intent(out) ::  de, dr
            use const_def, only: standard_cgrav
            ! Calculate Deltar (infall distance due to aerodynamic drag)
            ! We consider cross section area of planet.
            ! For compact objects (e.g. NS) we would need to use accretion radius.
            ! See e.g. equation B.3 in Tylenda & Soker 2006
            cdr = area*sqrt(standard_cgrav*m1) / m2
            dr = cdr * rho * sqrt(r) * dt
            ! Calculate DeltaE (energy loss)
            ! See e.g. equation B.2 in Tylenda & Soker 2006,
            ! where we have used v = keplerian velocity
            cde = (0.5d0*area)*(standard_cgrav*m1)**1.5d0
            de = cde *  rho * r**(-1.5d0) * dt
end subroutine what_a_drag
```



Planetary Disruption

When the ram pressure of the flow facing the planet is high enough to overcome the gravitational binding energy of the planet (Jia & Spruit 2018)

$$f \equiv \frac{\rho_{\rm ext} v^2}{\bar{\rho}_{\rm p} v_{\rm esc}^2} \approx 1$$

```
real(dp) function check_disruption(m_planet,r_planet,v_planet,rho_ambient) result(f)
real(dp), intent(in) :: m_planet,r_planet,v_planet,rho_ambient
real(dp) :: v_esc_planet_square, rho_planet
            rho_planet = 3d0*m_planet/(4d0*pi*pow3(r_planet))
            v_esc_planet_square = standard_cgrav*m_planet/r_planet
           f = (rho_ambient*pow2(v_planet)) / (rho_planet*v_esc_planet_square)
            ! Eq.5 in Jia & Spruit 2018,  destruction for f > 1
end function check_disruption
```