MESA summer school 2021

Morgan Chidester, Meridith Joyce and Alejandra Romero

Abstract

During this lab, we will use MESA to explore the evolution of a white dwarf (WD) during the cooling sequence. Each exercise will use a model that has already been evolved and cooled on the WD cooling track to a WD at 60 000 K.

The lab has 3 parts, WD Minilab 1, WD Minilab 2, and WD Maxilab. You will download two separate working directories from Dropbox: `WD_minlab_1.zip` and `WD_maxilab.zip`. The first is for both minilabs (combined), and the second is for the maxi lab. WD Minilab 2 builds upon WD Minilab 1, so be sure to save your data from WD Minilab 1. Each of the two directories contains different versions of files with the same names, so keep them separate!

1 WD Minilab 1

In the first lab, we will compute the evolution of a white dwarf from an initial model at the beginning of the cooling track. We will not consider diffusion processes due to computation time, but diffusion will not considerably affect the cooling times in this experiment.

1. From the Dropbox, download the `WD_minilab_1.zip` file. **Save this file outside of your main MESA installation directory.** Issue the command

   ```
   unzip WD_minilab_1.zip
   ```

   to unpack the directory into a folder called `WD_minilab_1`. Then, `cd WD_minilab_1/`

2. Set your MESA environment variables `MESASDK_ROOT` and `MESA_DIR` and issue the source command:

   ```
   source $MESASDK_ROOT/bin/mesasdk_init.sh
   ```

   Get in the habit of doing this every time you use MESA.

3. Open the `inlist_project` file and uncomment the line containing `saved_model_name` in `&star_job` (Fortran comments are indicated by `!` at the start of the line). Point this parameter to either the 2 $M_\odot$ or 3 $M_\odot$ ZAMS mass model you were assigned. **You will stick to this mass for the rest of your WD experiments.** Click here for help.

4. To build your executeable in the `WD_minilab_1` directory, issue the commands

   ```
   ./clean
   ./mk
   ```

5. Run the code:

   ```
   ./rn inlist_project
   ```

   to get a feel for a basic WD evolution.

   A `pg_plot` window containing three panels should pop up on your screen. Make a note of things you notice about the abundance plot, burning plot, and HR diagram. (i.e. What main isotopes is the star made of? Where is the model located in the HR diagram? etc.) The run should take ∼3 min to go from 60,000 K to when $\log(L/L_\odot) < -3.0$, which is the current termination condition.

6. When the run has finished, open and inspect the `LOGS/history.data` file. We want to add all sources of energy (both positive and negative) to this file. These include:

   - nuclear reactions from H burning: pp and CNO burning (this should be important at the beginning of the cooling evolution);
   - neutrino losses (from both nuclear and thermal neutrinos);
   - release of gravitational energy due to contraction (just think gravitational potential energy);
   - crystallization/sedimentation of heavy elements

   HINT 1: You will need to add quantities to your `history_columns.list` file.
   HINT 2: All the following parameters can be found by viewing and searching key terms in

   ```
   $MESA_DIR/star/defaults/history_columns.list
   ```

   Open the version of `history_columns.list` included in the `wd_minilab_1` directory to find prompts indicating where to insert the parameters. When you finish editing the file, re-run the code to see these new parameters listed in the `LOGS/history.data` file.

   Click here for solutions.

7. Make a plot of the temporal evolution of each of the quantities listed above. You can do this by editing `inlist_pgstar`. The `History_TrackX` panels will show you the evolution of these quantities individually, but you can set

   ```
   Grid2_win_flag = .true.
   ...
   Grid2_file_flag = .true.
   ```

   in `inlist_pgstar` to view and save the plots for all of these quantities in a single figure.

   The top line of `inlist_pgstar` determines whether the grid panel will appear during the run, and the second line dictates that the files will be saved in a directory called `grid_energy_sources_png`.

   Re-run your model with the updated `inlist_pgstar` and you should see these plots during runtime. Execution should take no longer than 5 min.

   In which part(s) of the evolution are the various energy sources important?

8. **[Time permitting]** Let's play with some stopping conditions:

   - Make your model terminate when it reaches $T_{eff} = 15\,000$ K. In `inlist_project`, search for the prompt

     ```
     !    ADD TEFF LOWER LIMIT HERE
     ```

     Re-run the model (runtime should not exceed 3 min). **Write down the model's age and luminosity at the termination point.**
     Click here for solution.

   - Turn off one of the energy sources, e.g. neutrino losses or one of the nuclear reactions. As a stopping condition, use the `max_age` parameter. Set this to the terminal age from the previous step.
     Ex.) If my star reached 15,000 K at an age of 2 Gyr, and I wanted to to turn off all nuclear reactions, I'd add

     ```
     max_age = 2d9 ! 2 Gyr
     max_abar_for_burning = -1 ! shut off all reactions
     ```

     to the `&controls` section of `inlist_project`. The second of these conditions states that if $\bar{A}$ (mean atomic mass) is greater than the assigned value, suppress all burning. Hence, `max_abar_for_burning = -1` shuts off the entire network.

     TIPS: Don't overwrite your `LOGS` and directory! Add `log_directory = '<insert new name>'` to the `controls` section of inlist project to save these files in a separate directory. You may also want to change the names of `Grid1_file_dir` and `Grid2_file_dir` in `inlist_pgstar` to save the png images in a separate directory.

     Is there a difference in the effective temperature or luminosity when an energy source is turned off vs included? If so, why do you think that happens?
     Stop the evolution when nuclear burning is no longer significant.

     NOTE: When you finish, **be sure to comment out** `max_abar_for_burning = -1` **AND** point `log_directory`, `Grid1_file_dir`, and `Grid2_file_dir` back to their original locations. (You may also rename the folders if you wish).

     HINT 1: You will again modify your stopping condition in the `&controls` section of `inlist_project`.
     HINT 2: What are the units for nuclear burning? How would you denote a "significant" contribution using a nuclear burning threshold?
     Click here for solution.
     What is the luminosity when nuclear burning is no longer significant?

9. **BONUS:**
   Numerical parameters can affect the physical predictions made by simulations, sometimes causing unintended changes to the results. In practice, it is very important to make sure your model is *converged* before reporting scientific results.
   To demonstrate the significance of these effects, we will change some of the convergence parameters in our inlist.

   - Locate the parameters `mesh_delta_coeff` and `time_delta_coeff` in your inlist. In which section are they located? In the first row of a table like Table 1, record their values along with the age and luminosity at the last time step of the model you have run most recently. Also include a rough estimate of the duration of that run (i.e. 3 minutes, 10 minutes, 4 seconds, etc.).

     Keeping all other inlist conditions the same, fill out subsequent rows of this table by recording the values of Age, LogL, and $t_{\rm run}$ when

     (a) `time_delta_coeff` $= 5$, `mesh_delta_coeff` $= 5$;
     (b) `time_delta_coeff` $= 0.5$, `mesh_delta_coeff` $= 0.5$;

     Was your original model converged?

   - MESA's default version of `history_columns.list`, which includes all parameter possibilities, can be found at

Table 1: Effects of Convergence Parameters (Mini Lab 1, Exercise 7)

Age (Gyr)	LogL	`time_delta_coeff`	`mesh_delta_coeff`	$t_{\rm run}$
2.0	0.5	1.0	1.0	4 min
??	??	0.5	1.0	?? min
...				

`$MESA_DIR/star/defaults/history_columns.list`

along with similar dictionaries containing all options for profile columns, `&star_job` settings, `&controls` settings, etc.

Since you will need to inspect these often, it is sometimes convenient to copy these files into your working directory directly, e.g.
`cp $MESA_DIR/star/defaults/profile_columns.list .`
but BEWARE of overwriting customized versions included with the labs!

Locate the parameter `var_control_target` in `controls.defaults`. Add this parameter to your inlist. Which value of `var_control_target` is being used in your run if it is **not** set it in your inlist?

Reset `mesh_delta_coeff` and `time_delta_coeff` to their original values. Increase the default value of `var_control_target` by an order of magnitude. Record Age, LogL, and $t_{\rm run}$ in your table. What does `var_control_target` do?

2 Minilab 2

NOTE:. If you made it through step 8 in Minilab 1, make sure to turn nuclear burning back on and ensure that `log_directory`, `Grid1_file_dir`, and `Grid2_file_dir` are pointing to the locations you want.

Use `run_star_extras` to compute the *cooling rate* and include this as a new output in the history file.

We will define the cooling rate as

$$\text{cooling\_rate} = \frac{T_{\text{eff},i} - T_{\text{eff},(i-1)}}{dt} \tag{1}$$

where $T_{\text{eff},i}$ is the effective temperature at the current time step, $T_{\text{eff},(i-1)}$ is the effective temperature from the previous timestep, and dt is the current timestep.

To set up your `run_star_extras`, do the following:

1. navigate to `<your_working_directory>/src/` and open `run_star_extras.f90`
2. replace the line
   `include 'standard_run_star_extras.inc'`
   with the contents of the file `$MESA_DIR/include/standard_run_star_extras.inc`
   You now have access to the subroutines that will allow you to define and print new quantities.
   Click here for emacs/vim help with inserting contents of a file.

   We will define our new quantities in terms of existing attributes of the `star_info` data structure. These are accessible via calls to the pointer s, as seen throughout the code. The star's effective temperature is accessed, for example, via the statement `s% Teff`
3. Define the new variables. For this calculation, we will need to define the cooling rate as well as $T_{\text{eff},(i-1)}$, since these are not existing attributes of `star_info`.
   Near the top of your `run_star_extras.f90` file, directly under the statement

   ```
   implicit none
   ```

   copy and insert the following:

   ```
   real(dp) :: Teff_prev
   real(dp) :: cooling_rate
   ```

   Now, the subroutines "know" about these quantities.
4. Assign a value to `Teff_prev` ($T_{\text{eff},(i-1)}$ in equation 1). At any timestep, we have access the current value of T_{eff}, but we must know the previous value simultaneously. As such, we must assign `Teff_prev` before the current time step is executed and store that value. We will do this in the `extras_start_step` subroutine, which is called before the time step has occurred.

   Navigate to the line

   ```
   integer function extras_start_step(id)
   ```

   which begins the `extras_start_step` function. Copy and paste:

   ```
   Teff_prev = s% Teff
   ```

   and insert this as the **last line** in the subroutine. It should go directly above

   ```
   end function extras_start_step
   ```

   We have now stored a value for $T_{\text{eff},(i-1)}$.
5. Define the cooling rate. Since we need the current value for this quantity, we need to navigate to a function called after the time step has been executed. We will move to `extras_finish_step`.

   Navigate to the line

```
integer function extras_finish_step(id)
```

and insert the cooling rate formula:

```
cooling_rate = (s% Teff - Teff_prev) / s% time_step ! K/year
```

immediately after the line

```
extras_finish_step = keep_going
```

Here, `s% Teff` grabs the current $T_{\rm eff}$ value ($T_{{\rm eff},i}$), `Teff_prev` is the value we stored in `extras_startup` ($T_{{\rm eff},(i-1)}$), and `s% time_step` is the change in time over the previous iteration, in years. You may also use, e.g., `s% dt` for the time step in seconds.

To write this rate out to the terminal, copy and paste:

```
write(*,*) 'COOLING RATE in K/yr :', cooling_rate
```

directly under the cooling_rate definition.

6. To write the `cooling_rate` as a column in the `history.data` file, navigate to the function `integer function how_many_extra_history_columns`. Change the value of `how_many_extra_history_columns` from 0 to 1, since we want to add 1 extra column.

   Move to the very next subroutine, `subroutine data_for_extra_history_columns`. Above the last line in that routine,

   ```
   end subroutine data_for_extra_history_columns
   ```

   copy and paste:

   ```
   names(1) = 'cooling_rate' !K/sec or K/yr
   vals(1)  = cooling_rate
   ```

   The first assignment tells `MESA` to set the column name as 'cooling_rate' (do not forget the quotation marks!). The second sets the quantity to be recorded.

7. **Recompile!** Recall that any time you modify `run_star_extras.f90`, you must recompile. Type `./mk` in your terminal. If this fails, run `./clean; ./mk`. If this compiles without error, run `./rn inlist_project`!

8. Confirm that `history.data` contains your new column for `cooling_rate`.

BONUS

You may have noticed that the subroutine `data_for_extra_history_columns` is "aware" of the `cooling_rate` quantity and its value despite the fact that `cooling_rate` is not defined within that subroutine. This is because the call to `extras_finish_step`—where `cooling_rate` *is* defined—is executed **before** the call to `data_for_extra_history_columns`.

Move the `cooling_rate` definition from `extras_finish_step` to the subroutine `extras_after_evolve` instead. What happens to the `cooling_rate` column in `history.data`? What does this tell you about the order in which the subroutines are run?

3 Maxilab: Changing the mass of the hydrogen envelope

There are many uncertainties surrounding the hydrogen mass left on the outer layers of a white dwarf, coming from uncertainties in the mass loss processes, interaction history, and residual nuclear burning. Based on single star evolution, there is an upper limit for the hydrogen mass given by nuclear reactions that consume hydrogen until the envelope is too thin and the burning conditions at the base of the envelope (pressure and temperature) are no longer maintained. The upper limit is dependent on the total mass of the white dwarf, being higher for lower stellar masses (Romero et al. 2019). However, there is no limit to the minimum amount of hydrogen that can remain in the envelope of a white dwarf. Since hydrogen is opaque, its abundance will influence the cooling time. The radius of the white dwarf will depend on the amount of hydrogen as well.

In `MESA`, the envelope mass is defined by:

`envelope_mass = star_mass - he_core_mass`

(see `$MESA_DIR/star/defaults/controls.defaults`)

We can thus change the envelope mass by changing either the star's mass or its He core mass.

In this lab, we're going to use the `mass_change` control, which strips off the star's outer layers. This will change the star's mass until a user-specified condition is reached. For this lab, that condition will depend on the hydrogen envelope.

We will implement a solution utilizing both the `inlist_project` and `run_star_extras.f90` files for practice, but **a simpler, alternative solution can be found here.** The alternate solution accomplishes the same thing, but it numerically relaxes the mass downwards, by stripping the envelope, before the evolution even begins.

To study the impact of the different hydrogen envelope masses on the cooling rate, we will make a grid. Choose a configuration (stellar mass and fraction of the hydrogen envelope) from this spreadsheet (click the Romero sheet).

You will record the cooling times at the specified $T_{\rm eff}$ values for a given fractional amount of the hydrogen envelope. You can do this by scrolling through the `LOGS/history.data` file to find $T_{\rm eff}$ values near 30000 K, 20000 K, 15000 K, etc. (values do not need to be exact!) Then, find the corresponding `cooling_rate` on the same line.

Be sure to navigate to the correct 2 or 3 $M_\odot$ sheet for the model you are using!. Replace **student X** in the column title with your name.
Note: You can write a script to navigate through the `LOGS/history.data` file or investigate the file manually.

3.1 Setting up the inlist

1. Download the `WD_maxilab.zip` file from the Dropbox and issue the command
   `unzip WD_maxilab.zip`

2. cd into the directory, set your `MESA_DIR`... etc., environment variables, and open the `inlist_project` file.

3. Uncomment (near top of the file) the line corresponding to the saved model you used in the past minilabs. (Either the 2 or 3 $M_\odot$ ZAMS models).

4. Under the appropriate prompt in the `&controls` section, add the lines

   ```
   mass_change = -1d-9 ! dm/dt in msun/yr
   x_logical_ctrl(1) = .true.
   ```

   A negative value for `mass_change` will decrease the mass. The choice of `mass_change = -1d-9` strips mass quickly enough to maintain the "desired" H envelope throughout most of the evolution without creating hydro-solver issues.

   The `x_logical_ctrl(1)` control tells `run_star_extras` to access a user-defined variable, defined in the next step.

5. Next, set the fraction of the H envelope you were assigned. Under the associated prompt, write

   ```
   x_ctrl(1) = <enter fraction> ! 0.5, 0.1, 1.0 etc.
   ```

   When `run_star_extras` is called, it will use this variable.

3.2 Setting up run_star_extras

1. Navigate to `src/run_star_extras.f90` file and open the file. Because the run strips mass during the evolution, we must save the star's initial mass value, as well as its initial helium core mass value, to obtain the correct fractional value of the initial envelope.

   First, we need to define these variables. Near the top of the file, under `! DEFINE INITIAL MASS VALUES`, add the lines

   ```
   real(dp) :: init_star_mass
   real(dp) :: init_he_core_mass
   ```

2. Next, find `subroutine extras_startup(id, restart, ierr)`. This subroutine is a good place to set initial values because it is executed at the beginning of the evolve step. Under `! SET INITIAL MASS VALUES`, assign

   ```
   init_star_mass = s% star_mass
   init_he_core_mass = s% he_core_mass
   ```

3. Now we need to tell `MESA` when to stop shedding mass. Navigate to the function `integer function extras_finish_step(id)`. Under the prompt `! STOP SHEDDING MASS CONDITION`, write

   ```
   if ((s% star_mass-s% he_core_mass) < (init_star_mass - init_he_core_mass)*s% x_ctrl(1)) then
         s% mass_change = 0d0
         write(*,*)'Ok, shed envelope!'
   end if
   ```

   Let's walk through what this is saying. First, let's break apart the `if` statement.

   Recall that
   `envelope_mass = star_mass - he_core mass`.

   So the first chunk in the if statement, `(s% star_mass-s% he_core_mass)`, is just the current envelope mass in this timestep.

   The second chunk `(init_star_mass - init_he_core_mass)*s% x_ctrl(1))` is the initial envelope mass multiplied by `s% x_ctrl(1)`. What is the `s% x_ctrl(1)` value again? This is the fractional amount of the hydrogen envelope you specified in `inlist_project`!

   If the current envelope mass is less than the fractional amount of the initial envelope mass, we tell `MESA` to stop shedding the mass of the star by setting `s% mass_change = 0d0`. The write statement prints an alert to the terminal to let you know when the star has stopped shedding layers.

4. Next, make sure your edited `run_star_extras` compiles. (*Remember to move out of the `src` directory, e.g. `cd ..`) Type `./mk` to compile. If the compilation is successful, run the code!

   Run the evolution until $T_{\rm eff} = 10000K$ (Click here for Teff stopping condition reminder) and compare the age and luminosity at termination to those from the sequence computed using the canonical value of the envelope. (Recall your original values from Minilab 1)

4 Minilab 1 solutions

4.1 Mini 1.3 Solution

After opening `inlist_project`, look at the top of the file in the `&star_jobs` section. You should see these lines:

```
saved_model_name = 'MESA_School_2Msol_60000.mod'
!saved_model_name = 'MESA_School_3Msol_60000.mod'
```

The top line is already uncommented, so if you were assigned the 2 $M_\odot$ model, you don't need to do anything. If you were assigned the 3 $M_\odot$ model, place a `!` at the beginning of the first line, and delete the `!` in the second line. Click here to go back to Minilab 1 part 3

4.2 Mini 1.6 Solution

NOTE: Remember, all the following parameters can be found by viewing and searching key parameters in

```
$MESA_DIR/star/defaults/history_columns.list
```

The given `history_columns.list` file in `WD_minilab_1` has prompts of where to insert the needed parameters.

For **nuclear reactions**, we want the hydrogen burning reaction from the pp chain and CNO cycle. Add the following lines to your `history_columns.list` file:

```
power_h_burn  ! total power from PP and CNO, excluding neutrinos
pp            ! pp chain only
cno           ! cno cycle only
```

For **neutrino losses**, we may want to include both nuclear and thermal neutrinos, as well as see how each contribute. To include total, only nuclear, and only thermal, add these lines to your `history_columns.list` file:

```
log_Lneu        ! log10 power from neutrinos, nuc and therm (Lsun units)
log_Lneu_nuc    ! log10 power from neutrinos, nuc sources only (Lsun units)
log_Lneu_nonnuc ! log10 power from neutrinos, therm sources only (Lsun units)
```

For **gravitational contraction**, a little more digging is required, but we want the total gravitational potential (U) energy. The log parameter serves best when plotting. Add this line to your `history_columns.list` file:

```
log_total_gravitational_energy !log(abs(total(U)))
```

For **crystallization/sedimentation**, add this line to your `history_columns.list` file:

```
total_WD_sedimentation_heating
```

After adding these to `history_columns.list`, re-run the code (`./rn inlist_project`).
Click here to go back to Minilab 1 part 6

4.3 Mini 1.8 Solution

The control that causes the model to terminate when $T_{\rm eff}$ drops below 15,000 K is:

```
Teff_lower_limit = 15000.0
```

Click here to go back to Minilab 1 part 8

4.4 mini1.8 Solutions

Any burning that is happening is most likely Hydrogen on the WD surface. So we want to stop when this burning is insignificant by setting some lower limit on hydrogen burning. Place the following limit in `inlist_project`:

```
power_h_burn_lower_limit = 1e-3
```

The run will terminate when H burning drops below this value. (**Note:** 1e-3 is given here, but the value is arbitrary; really any small value will do.) Click here to go back to Minilab1 part 8

4.5 Minilab 2 emacs/vim easy help

If using emacs or vim, simply delete the line `include 'standard_run_star_extras.inc'` and type:

```
C-x i <filename>
```

for emacs or

```
:r <filename>
```

for vim.

For both, `filename` is just `$MESA_DIR/include/standard_run_star_extras.inc`.. You can of course just copy and paste as well :)

Click here to go back to Minilab 2

4.6 Maxilab simple solution

The simplest solution is to use the `relax_mass` and `new_mass` controls in `inlist_project`. Open `inlist_project` and under `&star_job`, put

```
relax_mass = .true. ! gradually change total mass
new_mass = <enter new mass in Msun units>
```

Where you enter the new mass that your model would have after shedding the desired amount of the Hydrogen envelope. If my model started at 0.580 $M_\odot$ with a Helium core out to 0.578 $M_\odot$, then my initial H envelope would be 0.002 $M_\odot$. If I wanted one tenth of this, 0.0002 $M_\odot$, then my new mass would be 0.578 + 0.0002 = 0.57802 $M_\odot$. After inserting the new mass, `MESA` will relax the model to gradually shed the outer layers until the new mass is reached. It does this at the start of the run, before any steps are taken in the evolution.

Click here to go back to Maxilab

4.7 Maxilab Teff = 10 000 solution

The control that causes the model to terminate when $T_{\rm eff}$ drops below 10,000 K is:

```
    Teff_lower_limit = 10000.0
```

Click here to go back to end of Maxilab.