MESA Workshop: Notes

Mathieu Renzo
Università di Pisa
17-19 Settembre 2014

Abstract

We introduce the numerical methods and some example of the results of the open-source one-dimensional stellar evolution code MESA. Our aim is to improve the awareness of the approximations, the numerical techniques, and their limits, needed in the field of Stellar Astrophysics. The workshop includes a "hands-on" session, in which participants are invited to run some simulations and familiarize with the use of MESA. This notes are written for personal purpose, so if you find (and you probably will!) any error, write to mathren90@gmail.com.

Contents

1 Introduction

MESA (Modules for Experiment in Stellar Astrophysics) [1, 2] is an open-source one-dimensional stellar evolution code, organized as a set of independent threadsafe[1] modules. Each module can also be used separately by other codes, thanks to its "public"interface and the separate "private" implementation, which makes it easy to call from outside.

Each module deals with one aspect of the physics (e.g. EOS, opacities, the nuclear network, etc...) or the numerics (i.e. matrix operations[2], one-dimensional interpolation of data sets, error handling, etc...). We will focus the attention on the module `star` (see `$MESA_DIR/star`), which is the one that calls all the other modules in the right order to simulate a star.

The code is written in FORTRAN 90, and is designed to be easy to modify. Basically, anything can be modified, so in principle any user can adapt MESA to their own need. It is usually very easy to do so, even with very little FORTRAN knowledge, thanks to the use of the "hooks" in `run_star_extras.f` (see §3.3.1). These allows the implementation of customized wind scheme, stopping criterion, meshing algorithms, output settings, etc...

MESA's versatility makes it a powerful tool for Stellar Astrophysics: in principle MESA is capable of simulating any object in between planets and super massive stars (SMSs, $M \geq 10^4 M_\odot$). The caveat, as in any numerical simulation is: "is the result trustworth?..."

2 Numerical Methods

The equations describing a stellar structure are a set of coupled non-linear differential equations, which cannot be solved analytically. Therefore, the field of stellar astrophysics relies on numerical simulations, which requires a number of (numerical) approximations (e.g. finite differences instead of derivatives) to get a solution. MESA `star` is a Henyey-like [3] code, which uses the MESA modules to solve the equations of a stellar structure, with automatic mesh refinement, adaptive time-step control, and analytic jacobians. In the following sections we briefly describe the approximations used in MESA, as an example of what is common in stellar evolutionary codes. The reader is referred to [1, 2] for more details.

2.1 1D or 1.5D: Spherical Symmetry

MESA is a one-dimensional code, that is it assumes spherical symmetry and uses only one coordinate. This approximation is needed because of the prohibitive computational cost of full-three-dimensional simulation, and yet it is known to be poor.

For example, we know that stars rotate. In MESA it is possible to simulate some rotational effects in a parametric way, using the so-called "shellular approximation" [3], i.e. assuming that the frequency is constant on isobar surfaces. This is motivated by the absence of restoring forces along the isobar surfaces (i.e. in the horizontal direction): differential rotation in

[1] Multicore processors capabilities can be exploited using the OpenMP parallelization.
[2] Sparse matrix operations are handled with the "SPARSEKIT" solver, see [1].
[3] Sometimes it is indicated as "1.5D".

between radiative regions would cause turbulence in the horizontal direction, the turbulence would then erases the angular velocity gradients [2], imposing a constant frequency on isobar surfaces.

Even if we limit to the study of non-rotating stars, there are still problems: convection is intrinsically a three-dimensional problem, it involves turbulent cascades, fluid vorticity, etc... The convection is treated with the use of the mixing length theory (MLT), yet another parametric model, which is likely to cause very large systematic errors, and this requires also the overshooting regions to account for the convective bubbles deceleration, so even more parametric models. It is not our purpose to cover in details these parametric models, which should be known from the stellar astrophysics course. For more details any stellar astrophysics book can be used as reference, see e.g. [3].

2.2 Meshing

In order to be able to solve differential equations (such as those describing a stellar structure), computers need to discretize them adopting a finite-difference scheme. At the beginning of each step, MESA discretize the stellar structures, i.e. it defines a grid of points defining cell boundaries, and evaluates finite differences between the adjacent points instead of derivatives,

$$\frac{df}{dm} \rightarrow \frac{f(m_{k+1}) - f(m_k)}{m_{k+1} - m_k} \quad . \tag{1}$$

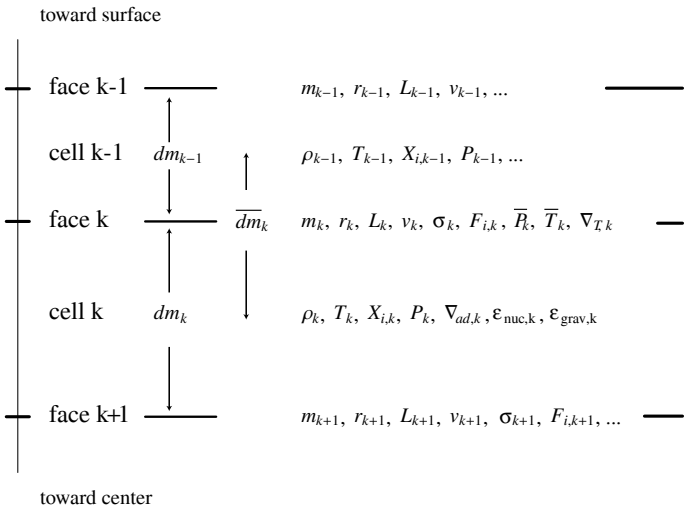

Figure 1: Sketch of MESA mesh structure, from Paxton *et al.* [1]. Note that cells are numbered from 0 on the surface, to the maximum value in the center.

The number of points is determined automatically by full-filling some user defined conditions. In a typical situation, the user controls the number of points through a multi-

plying factor on the maximum relative gradient allowed between two adjacent cells (using `mesh_delta_coeff` and `mesh_delta_coeff_for_highT`). If MESA finds a relative (spatial) variation between two adjacent cells larger than what the user defines, it splits the largest of the two cells. If this is not sufficient to full-fill the user needs, it then crashes. The maximum number of cell produced by the splitting is two by default, but it can be increased.

Note that changing the spatial resolution can strongly affect the results: for example if the core is under-resolved, the temperature of the central cell will be an average of a large region in the core, thus it could result in being too low for the ignition of the next fuel specie.

In a typical situation, MESA uses a lagrangian (i.e. bounded to the matter) mesh adopting the mass included in the outer boundary of the cell m as independent coordinate. This can change for some application, such as accretion of mass: in this case, MESA uses the fractional mass $q = m/M$ as variable in the outer portion of the star (which is not a lagrangian coordinate if $\dot{M} \neq 0$) and the usual m in the center, where no new mass is deposited. In such case there is a overlapping region between the two mesh, to allow a smooth conjunction.

2.3 Timestep control

Not only the spatial dependence must be discretized, but also the temporal evolution. At the end of each model computation, MESA proposes a timestep for the next iteration. The timestep Δt is the amount of time we allow to elapse to compute time-derivatives in a finite differences scheme

$$\frac{df}{dt} = \frac{f(t_{k+1}) - f(t_k)}{t_{k+1} - t_k} \rightarrow \Delta t_{k+1} = t_{k+1} - t_k \ , \tag{2}$$

thus, it should be sufficiently small to resolve all time variations, but sufficiently large to allow the simulation of a star in a reasonable run-time. This makes the selection of the timestep a complex process, since in stars there are several timescales involved (nuclear, free fall, Kelvin-Helmholtz, mass change, etc...) and they are usually very different from each other.

MESA does the timestep selection in a two-stage process. First, Digital Control Theory is used to propose a timestep. This means that Fourier transform are used to evaluate the characteristic frequencies of the system, and a timestep smaller than all the corresponding period is picked. At this stage MESA only checks if the control variable v_c, an unweighted average of the relative variation of $\log(T)$, $\log(\rho)$ and $\log(r)$ in all cells is smaller than a user-defined value `varcontrol_target`, whose default is $v_t = 10^{-4}$. The timestep proposed during this stage is evaluated as a function of the previous two timesteps, to improve numerical stability. If Δt_{k-1}, and Δt_k are the two previous timesteps, $v_{c,k-1}$ and $v_{c,k}$ their control variables, then the new timestep to be determined is

$$\Delta t_{k+1} = \Delta t_k g \left(\frac{g(v_t/v_{c,k}) g(v_t/v_{c,k-1})}{g(\Delta t_k/\Delta t_{k-1})} \right)^{1/4} \ , \tag{3}$$

where $g(x) \stackrel{\text{def}}{=} 1 + 2\tan^{-1}(0.5(x-1))$.

Then, in the second stage, additional controls are performed and can cause an additional timestep reduction. This include limit on the maximum nuclear burning rate, maximum change in surface luminosity, etc... All settings can be tuned by the user.

2.4 Reformulation of the Stellar Structures Equations for Numerical Stability

In order to improve numerical stability, the stellar structures equations that MESA has to solve are reformulated on such grid as follows. In general extensive quantities (r, L, m, etc...) are given at the cells outer boundary, while intensive quantities (T, ρ, P, etc...) are cell mass averages. Let us begin with the hydrostatic balance equation,

$$\frac{P_{k-1} - P_k}{0.5(dm_{k-1} - dm_k)} = \left(\left.\frac{dP}{dm}\right|_k\right)_{\text{static}} + \left(\left.\frac{dP}{dm}\right|_k\right)_{\text{dynamic}} = -\frac{Gm_k}{4\pi r_k^4} - \frac{a_k}{4\pi r_k^2} \ , \tag{4}$$

where P_k is the (cell average) pressure, dm_k is the cell mass, r_k its radius and

$$a_k \stackrel{\text{def}}{=} \frac{dv_k}{dt} \ , \ v_k = r_k \frac{d\ln(r_k)}{dt} \ , \tag{5}$$

is the lagrangian acceleration, with v_k the cell velocity computed as the variation of the radius between two successive timesteps. Note the use of logarithmic function to have smaller numbers to deal with and reduce the round-off error. Eq. (4) includes both the hydrostatic and the so-called "dynamical pressure"[4] term, which are the first and second term on the right hand side, respectively. If the flag to trace velocities is not true, the latter term is set to zero, otherwise, to help numerical stability, v_k is divided by the local sound speed.

The continuity equation is formulated in a rather strange way, to improve the round-off errors:

$$\ln(r_k) = \frac{1}{3}\ln\left[r_{k+1}^3 + \frac{3}{4\pi}\frac{dm_k}{\rho_k}\right] \ , \tag{6}$$

where ρ_k is the average density in the cell. The energy transport equation is formulated as

$$\frac{T_{k-1} - T_k}{0.5(dm_{k-1} - dm_k)} = -\nabla_{T,k}\left(\left.\frac{dP}{dm}\right|_k\right)_{\text{static}} \frac{< T_k >}{< P_k >} \ , \tag{7}$$

where T_k is the cell temperature, and

$$< f_k > \stackrel{\text{def}}{=} \frac{f_{k-1}dm_k + f_k dm_{k-1}}{dm_{k+1} + dm_k} \ , \tag{8}$$

is an average value of f(=T or P) between the two adjacent cells. For numerical stability Eq. (4) and Eq. (7) are multiplied by $< P_k >$, and $< T_k >$ respectively.
Energy conservation is expressed as:

$$L_k - L_{k+1} = dm_k\{\varepsilon_{\text{nuc}} - \varepsilon_\nu + \varepsilon_{\text{grav}}\} \ , \tag{9}$$

where ε_i are the energy generation rate per unit mass:

- ε_{nuc} are taken from the NACRE or Jina databases, as tables in function of T and ρ. MESA does the interpolation between the available data automatically. Usually in ε_{nuc} the energy going into neutrinos from nuclear reactions is already subtracted;

[4]The "dynamical pressure" is not really a pressure, but a correction to account for the movement of mass. It is usually a very small term.

- ε_ν is the thermal neutrino losses - significant for the late burning stages in massive stars. This term includes only thermal neutrinos because the nuclear neutrinos are already subtracted from the nuclear energy generation rate;
- $\varepsilon_{\rm grav} = -Tds/dt$ is the so-called "gravitational work". The time derivative of the specific entropy s is evaluated as a function of the thermodynamical quantities, see Eq. (12) in [1].

Finally we have the equation for the chemical abundance for the $i-$th specie in the $k-$th cell, namely

$$X_{i,k}(t+\delta t) = X_{i,k}(t) + \left(\frac{dX_{i,k}}{dt}\right)_{\rm nuc} \delta t + (X_{i,k} - X_{i,k-1}) \frac{\sigma_k \delta t}{0.5(dm_{k-1} - dm_k)} \tag{10}$$

where the second term on the left hand side is evaluated from the nuclear tables, and the third (diffusive) term includes all mixing processes (chemical diffusion, radiative levitation, overshooting, shear, etc...) summed[5].

Note that MESA only traces the abundances of the isotopes in the chosen nuclear network (usually they are only enough to generate the energy necessary to sustain the star through their fusion, i.e. $\sim$21 isotopes). The nuclear network is automatically built by listing the isotope wanted: MESA looks in the nuclear database specified (NACRE of Jina) and include all the reaction linking the isotopes listed. Otherwise, the user can also specify the list of reactions. Once the nuclear network is built, MESA uses the Bulirsch-Stoer method [4] to compute the abundances: the function $X_{i,k} \equiv X_{i,k}(t)$ is supposed to be analytic and the timestep Δt is divided in a (not known *a priori*) number of substeps, $X_{i,k}$ is evaluated at each substep and then fitted. This allows to simulate complex nuclear burning, such as Si burning (i.e. Quasi Statistical Equilibrium regime), without introducing the equilibrium approximation.

Note that, since nuclear reactions involve usually only 2 nuclei, most of the terms for nuclear reactions on the right hand side of Eq. (10) involve only two isotopes. Thus the matrix to solve,

$$\begin{pmatrix} X_{^1H} \\ X_{^2H} \\ \vdots \end{pmatrix}_{t_{k+1}} = \begin{pmatrix} R_{^1H,^1H} & \cdots \\ \vdots & \ddots \end{pmatrix}_{t_{k+1}} \cdot \begin{pmatrix} X_{^1H} \\ X_{^2H} \\ \vdots \end{pmatrix}_{t_k} \tag{11}$$

to get the updated abundances from the old, has a lot of zero entries and MESA takes advantage of sparse matrix solvers to reduce the run time.

The initial abundances are specified through X, Y, and Z, and a scaling with the solar value is assumed for each isotope. This assumption is typical in stellar astrophysics codes, but nevertheless it is rather poor: we have evidences that isotopes are built in different places in the universe (Novae, Supernovae, binary neutron star mergers, etc...) and in principle nothing assures that the scaling between isotopes is the same in the Sun and everywhere else.

[5]The use of a "diffusive" approximation for dynamical phenomena is usual but problematic in many stellar evolution codes. Note also that instabilities could cancel out, instead of adding the user can also specifically list the reactions. to each other

2.5 Algorithm

The set of equation is simultaneously[6] solved for the structure (T, ρ, L,etc...) and composition ($X_i \; \forall i$ in the nuclear network).

In order to perform the integration of the equations Eq. (4), Eq. (6), Eq. (7), Eq. (9), Eq. (10), and the (tabulated) EOS $T \equiv T(\rho, P)$, MESA uses a Newton-Raphson one-dimensional solver[7], represented by

$$0 = \mathbb{F}(y) = \mathbb{F}(y_i + \delta y_i) = \mathbb{F}(y_i) + \left[\frac{d\mathbb{F}(y)}{dy}\right]_i \delta y_i + \mathbb{O}((\delta y_i)^2) \;\; , \tag{12}$$

where y_i is a trial solution (usually, the first iteration uses the solution of the previous timestep, just re-meshed and with the mass change applied, as first guess), $\mathbb{F}$ is the residual, i.e. the distance between the trial solution y_i and the (unknown) solution, and the term in brackets is the Jacobian matrix, calculated analytically. For the calculation of the Jacobian matrix, each MESA module evaluates not only the quantities it deals with, but also all the partial derivatives of them with respect any input variable.

In a Henyey-like code the solution for the stellar structure at a given time is found starting from a trial solution y_i, and varying all the parameters at all the points until the required degree of accuracy is reached. This is particularly convenient for problems, such as the set of equations for a stellar structure, in which boundary conditions are given at both ends of the domain, i.e. in the center - T_c and ρ_c - and at the surface - R and L). In fact, the Henyey method (in contrast with the shooting method) varies also the boundary condition, and finds more easily a solution within the degree of accuracy required [3].

Fig. (2) and Fig. (3) show the temperature as a function of mass, at each iteration of the Newton-Raphson numerical solver for two stars at the beginning of the main sequence, of 22.5 $M_\odot$ and 1.4 $M_\odot$, respectively. Note the use of a linear scale on both axis: the relative adjustment of each quantity are small and not appreciable in a logarithmic scale. These two models are computed from the zero age main sequence (ZAMS), and the plots shown are for two models after the starting point. The very small corrections needed to get from the first guess of the solver to an accepted solution indicates that the initial model are already close to the solution. Notice that MESA can start from the main sequence using built in models of solar metallicity [1]. The metallicity can then be changed with an initial "pseudo-evolution" (i.e. evolution at fixed time). There are 32 initial models in the mass range [$0.8M_\odot$, $100M_\odot$], and MESA interpolates between those to create zero age main sequence stars of any mass. The other option to start a run is to use a homogeneous composition model, with a central temperature too low for any nuclear fusion to happen and evolve it through the pre-main sequence (pre-MS).

Fig. (4) is similar to Fig. (2) and Fig. (3), but for a $9M_\odot$ solar metallicity star 2 models after the terminal age main sequence (TAMS), i.e. central hydrogen depletion, $X_c < 0.01$. The initial model used here is computed with somewhat different parameters until TAMS and then restarted, nevertheless the convergence to an acceptable hydrostatic solution is rapid.

If MESA fails in finding a solution, before crashing it first does a "retry", i.e. restart from the previous model with a reduced timestep. If it cannot archive convergence again, it does

[6] Even if the user can decide to split the solution for the chemical composition. In this way the chemical mixing processes are done differently (how depends on the user choice), allowing a faster convergence.

[7] This algorithm was born to find the zero of a function with an iterative procedure, $x_{n+1} = x_n - f(x_n)/f'(x_n)$

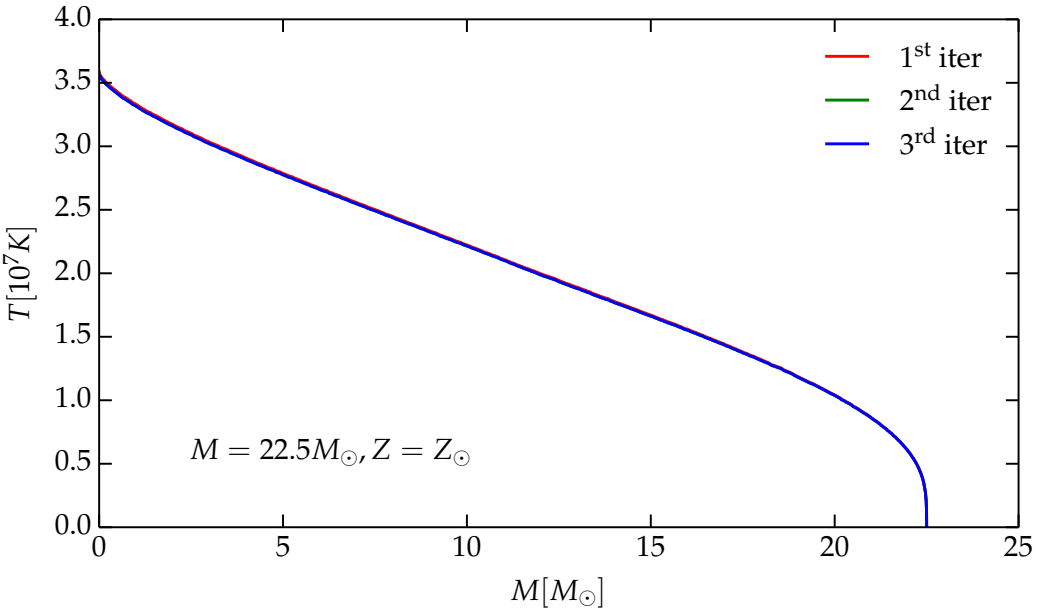

Figure 2: $T \equiv T(M)$ for a $M = 22.5M_\odot$ star, two models after the zero age main sequence.

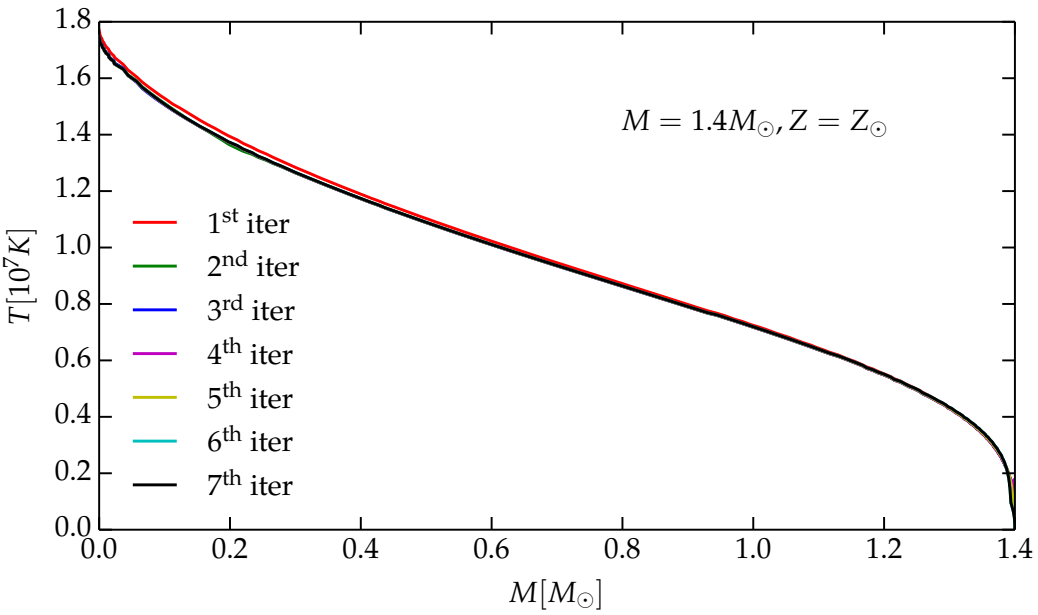

Figure 3: $T \equiv T(M)$ for a $M = 1.4M_\odot$ star, two models after the zero age main sequence.

a "backup", i.e. it goes two steps back and restarts from there with a reduced timestep.

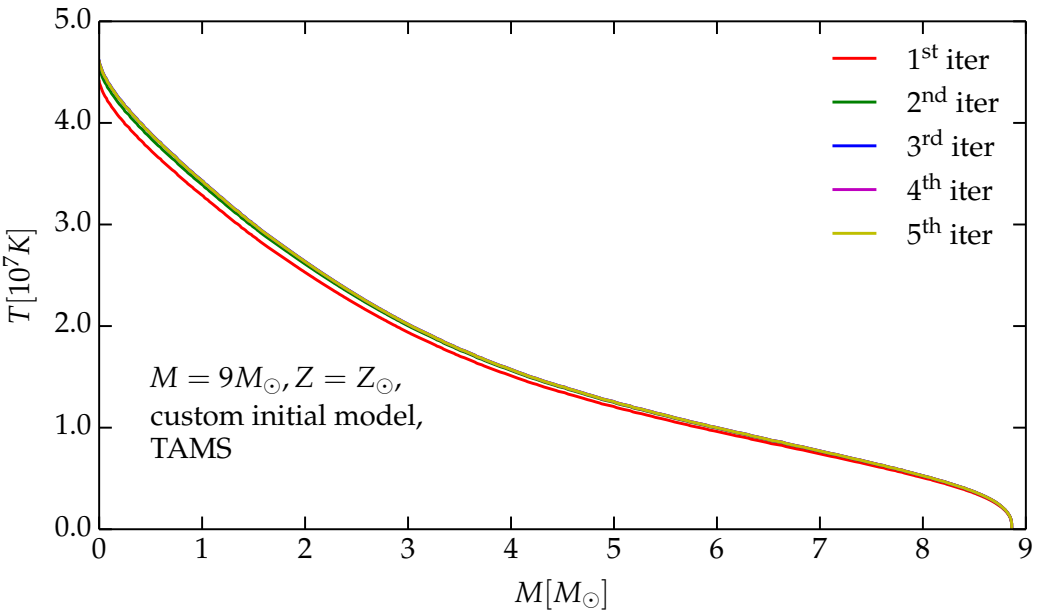

Figure 4: $T \equiv T(M)$ for a $M = 9M_\odot$ star, two models after the terminal age main sequence.

2.6 Run time

For many stellar astrophysics problems not involving extreme regimes (such as the late stages of nuclear burning, the Helium flash, etc...) MESA is a very efficient tool, thanks to the use of sparse Matrix solvers (i.e. solvers that take advantage of the large number of zero entries in matrices). For example, in Fig. (2.6) we compare the run time of three stars computed from ZAMS to TAMS. The run times are less than a minute in each case.

The more challenging phases of the stellar evolution are those requiring small timestep and/or large nuclear network. This will typically happen in unstable regimes (e.g. super-Eddington luminosity), or when the nuclear rates are very high and many isotopes are involved (such as Si burning).

2.7 Parametric models: the example of massive stars

Despite the fact stellar evolution is a well established field, with high precision predictive capabilities, it still relies on one dimensional simulations, which require many parametric and approximated models (such as MLT, shellular approximation, wind mass loss, etc...). Very often these models require the use of fudge factors to "artificially" enhance or suppress some effects (e.g. the wind efficiency η). The introduction of such models could affect the evolutionary models in a not well understood way: often there is no way around them, so there are no models avoiding the use of arbitrary choices to use as comparison. This is particularly true for the modeling of massive stars: we use them as an example of the rough[8] approximations needed to simulate a star until core collapse.

[8]Here "rough" just means: "it is the best we can do, even if we know it is wrong".

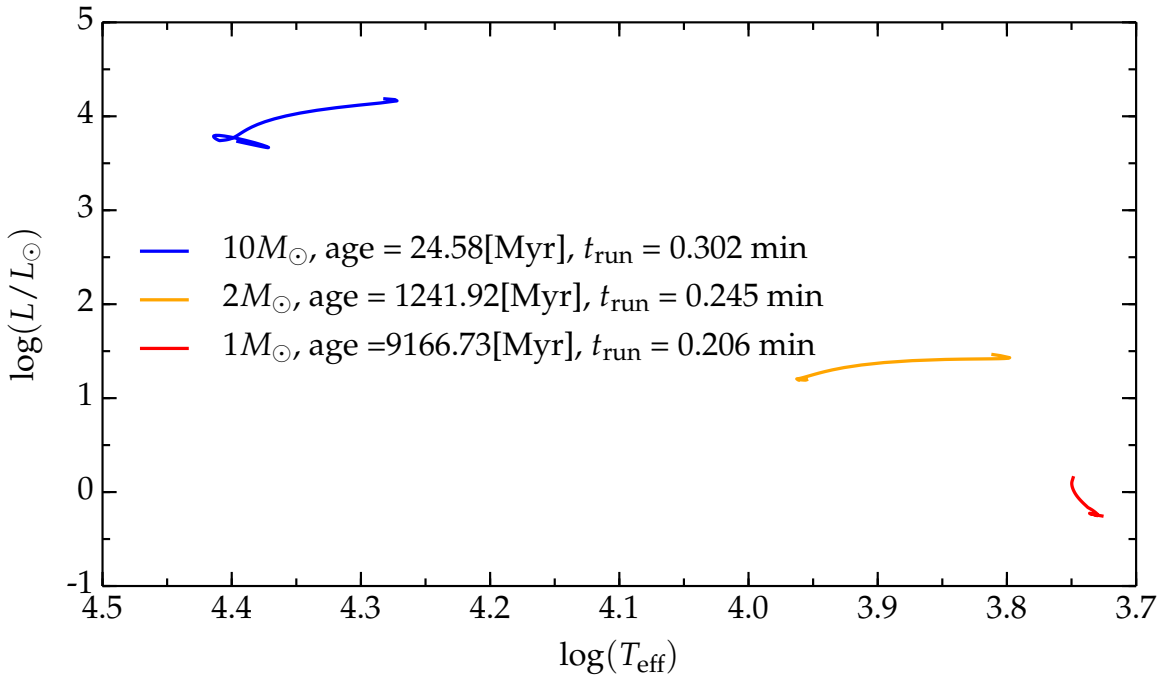

Figure 5: Comparison between the run time for the main sequence evolution of three solar metallicity stars with different $M_{\rm ZAMS}$

Massive stars model have two main issues: late nuclear burning and super-Eddington envelopes.

The first one is a pure computational issue[9]: the late burning stages are complex because of the interplay between the (thermal) neutrino cooling (dominant after O ignition because of the high temperatures reached) and energy generation rate. This often causes the onset of many convective shells. Moreover, the last nuclear burning stage, Si burning, requires the inclusion of a very large number of isotopes in the network (Quasi Statistical Equilibrium regime, QSE): this increases dramatically the computational cost of this stage. As a work around for users not interested in the structure of the core, MESA can go through the Si burning stage using its default 21 isotopes network, generating the correct amount of energy to sustain the star, but faking all electron captures on one single isotope and producing only one neutron rich nuclear specie ${}^A_Z\tilde{X}$. The default "fake electron capture reaction" to simulate neutronization during Si burning is $2e^- + {}^{56}\mathrm{Fe} \rightarrow {}^{56}\mathrm{Cr} + 2\nu_e$, so ${}^A_Z\tilde{X} = {}^{56}_{24}\mathrm{Cr}$. This procedure reduces the number of electrons, but only until

$$Y_e \stackrel{\rm def}{=} \sum_{i=1} X_i \frac{Z_i}{A_i} \equiv Y_e({}^A_Z\tilde{X}) \neq \{Y_e(QSE), Y_e(\text{large net})\} \ , \tag{13}$$

i.e. the electron mole fraction becomes the one of the isotope produced by the fake electron capture reaction. This value is typically very different of what one would get with a QSE

[9] In the sense that a physical model is well established for the late burning stages.

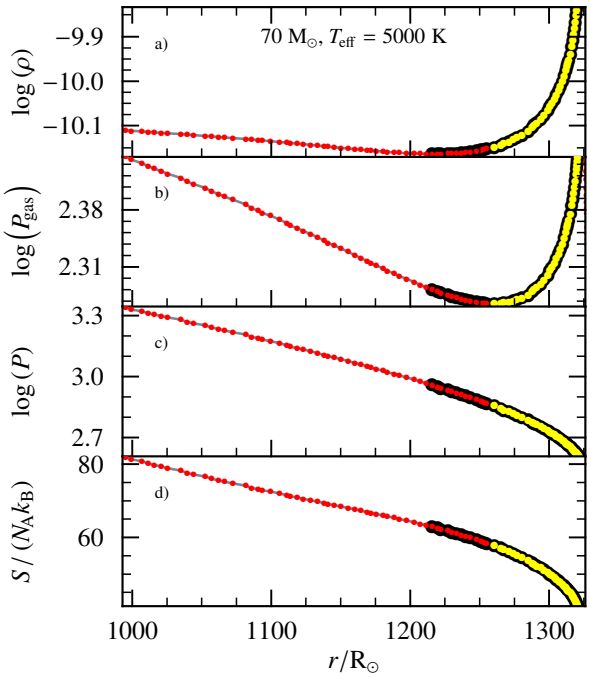

Figure 6: From [2], from bottom to top: specific entropy, total pressure, gas pressure and density profile in a super-Eddington envelope of a $70M_\odot$ star. The decrease ion entropy indicates the region is superadiabatic. Small red dots indicate region where $L_{\rm rad} \leq L_{\rm Edd}$, big red dots indicate regions where a density inversion is expected without a pressure inversion and finally yellow dots indicate the density and pressure inversion.

network or a network including hundreds of isotopes without equilibrium assumptions. This influences the Chandrasekhar mass of the core and its final structure. Anyway, this is not the only option available in MESA: the user can switch to a larger network for the final burning stage, but they have to pay the price of very large computational times.

The second issue, super-Eddington envelopes, is more severe, in the sense that we currently do not have a complete physical understanding of what happens. It arises only for stars more massive than $\sim 20M_\odot$, Fig. (7). In the inner envelope of these stars the luminosity L can exceed the Eddington luminosity,

$$L(r) > L_{\rm Edd}(r) \stackrel{\rm def}{=} \frac{4\pi GM(r)c}{\kappa(r)} \ , \tag{14}$$

usually because an increase of the opacity (e.g. Fe bump because of recombination), lowers the Eddington luminosity below the actual local luminosity. In such situation the region is dynamically (and numerically) unstable. If this happens in a region that happens to be convective, usually convection is unable to carry out energy in an efficient way (the convective flux is limited by $F_{\rm conv} \sim \rho c_s^3$, since the convective velocity must be less than the speed of sound c_s), and density and pressure inversion arise [2], Fig. (6). This forces the timesteps to assume prohibitively small values. This situation is thought to be a true physical instability (maybe triggering LBVs mass loss [5]).

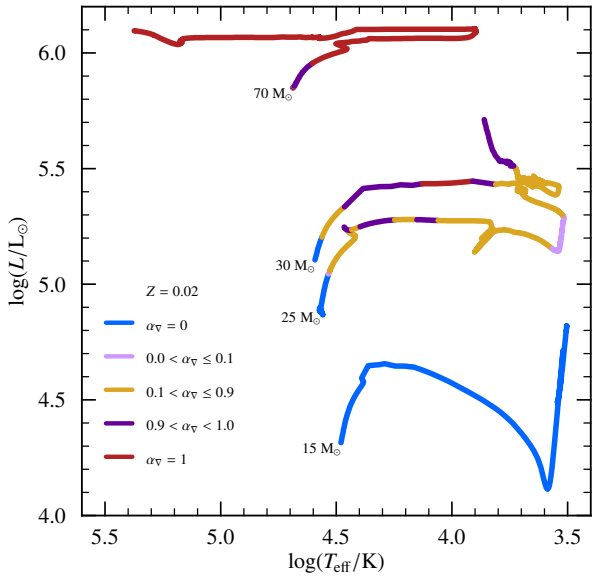

Figure 7: From [2], HR diagram for different massive stars. The colors indicate the values of α_∇ which is automatically computed at each timestep: the higher it is, the more necessary the MLT++ is.

To deal with super-Eddington envelopes in massive stars, MESA adopts the MLT++[10]: when a super-Eddington envelope forms (radiative dominated envelope) and the convective flux is not strong enough to carry away the energy, MESA artificially reduces the temperature gradient, killing by hand the superadiabaticity and forbidding the onset of convection. This of course affects the surface (i.e. observable) variables such $T_{\rm Eff}$, $L(r = R_*)$, etc...Fig. (8) More specifically, when the superadiabaticity grows beyond a given threshold, it is reduced by a factor $\alpha_\nabla f_\nabla$, where f_∇ is a user defined parameter and α_∇ is a sort of efficiency of the superadiabaticity reduction computed automatically at each timestep. Fig. (7) illustrates when the MLT++ comes in during the evolution of massive stars: the 15 $M_\odot$ star goes from ZAMS to the onset of core collapse without using it, while it is fully on for almost the entire evolution in the $70 M_\odot$ model.

The MLT++ can be turned on/off by setting the parameter `okay_to_reduce_gradT` to `.true./.false.` in the `controls` namelist §3.3, respectively.
See also `$MESA_DIR/star/defaults/controls.default`.

[10] Note that any one dimensional stellar evolution code has to use some "cheat" for this problem.

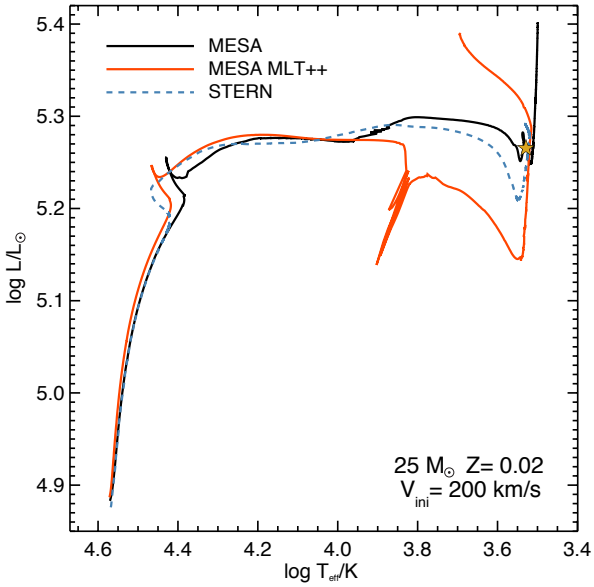

Figure 8: From [2], comparison between evolutionary tracks computed with MESA with and without MLT++ and STERN models.

3 Using MESA

In the following section a basic knowledge of bash is assumed.

3.1 Installation - (just a few references)

The installation of MESA on a Unix machine (including laptops!) should be straightforward using the software development kit `MESA SDK`, http://www.astro.wisc.edu/~townsend/static.php?ref=mesasdk. The advice here is to follow the instructions for the installation of `MESA SDK` first, and then install MESA. The reader is referred to http://mesa.sourceforge.net/ and material therein for the installation guide. Many useful tutorials are available also from http://mestastar.org/teaching-materials.

A typical installation on a Unix system will have the code in a folder whose path is saved in the global variable `MESA_DIR`. Its subfolders contain all the modules, with their public interface and private implementation. It is a good practice to search through the code using the command

```
grep -r ‘‘what I want to search’’ $MESA_DIR ,
```

which will search for the expression between quotation marks recursively in all files from the code directory[11]. Thus, it is very useful to search for a variable of which we know the name, routines, or comments in the code.

Another interesting place for a new user is `$MESA_DIR/star/test_suite`, where the work directories for a large set of test cases can be found. Given the variety of test cases available, it is very likely that a good starting point for any project sits in here.

[11] However, for faster searches it is advisable to use the relevant subfolder as the last argument, i.e. change `$MESA_DIR` in the path of the relevant subfolder.

3.2 A typical `work` directory

It is good practice to run simulations outside the code folder, to keep things in order and avoid damages. A sample of working directory is available in `$MESA_DIR/star/work`, we can copy it anywhere in our machine, customize it and start our new project. All the input and output files of our current project will be stored in here by default.

Inside the work directory we find the `LOGS` directory, which will contain the data output, in the form of ASCII text files:

- `profile*.data`: spatial slices of the star at a given time. What data set is saved can be customized copying `$MESA_DIR/star/default/profile_columns.list` and commenting out or uncommenting (the FORTRAN syntax for comments is " ! ") the relevant columns. If the data needed is not in the `profile_columns.list`, it can still be calculated and included using the `run_star_extras.f` (§3.3.1);
- `history.data`: this contains a header with some data of the entire star (such as initial mass, metallicity, etc...), and then a row for each model computed[12]. Once again, to customize the data set saved at each timestep, the user can copy `$MESA_DIR/star/default/history_columns.list` in the `work` directory and comment out or uncomment its columns; or if the data wanted is not evaluated by MESA, it can be included using the `run_star_extras.f` (§3.3.1).

Moreover, every now and then[13] MESA will save a so-called "photo", i.e. a binary file storing all the information regarding the run, in the folder `photos`. These photos can be used to restart the runs, and if the setting used is the same, the evolution is granted to be identical bit-to-bit. A MESA run can also be restarted from a "model" (if the `star_job` namelist dictates so), which is an ASCII text file. The difference between these two methods of saving data for a restart, is that the information stored in a photo is complete but there is no warranty of compatibility between different MESA versions, while a model stores much less information, but it is granted to be compatible with past and future MESA versions.

Another subfolder is `src`, which will contain the `run_star_extras.f` needed for more advanced modifications to MESA (see §3.3.1).

3.3 How to control MESA - the Inlists

The most important things in the working directory is the `inlist`, which is where MESA will look for instructions at the beginning of each run. It is organized in 3 parts (or `namelists`):

- `star_job`: it is used to set most of the initial conditions and settings, such as the nuclear network wanted, if we need to set special nuclear reaction rates, change the initial metallicity, load a previously computed model as a starting point, etc... Moreover, the user can decide here if they want to save the data of the last timestep, and/or create a model for further restarts. For more details the reader is referred to the comment-rich file `$MESA_DIR/star/defaults/star_job.defaults` ;

[12] When to write a line of the `history.data` file can be easily customized from the `controls` namelist, see below

[13] See `$MESA_DIR/star/defaults/controls.default` for the customization of when a photo is saved.

- `controls`: this contains the variables used at every timestep, such as the mesh and timestep criteria, wind mass loss, overshooting parameter, convection criterion, etc... Again, for more details see `$MESA_DIR/star/defaults/controls.defaults` ;
- `pgstar`: This controls the setting for the output plots produced during run time, including settings to save each frame to post-process them into a movie. It is the only namelist read at each timestep, instead of once at the beginning of the run.

Note that the user is allowed to include in their inlist only the variables they are changing: any not cited setting will be set to its default. This is intended to make the inlist more easy to read and interpret.

The default `inlist` file is just telling MESA to read other files. This allows the user to use separate files for each part, or even to load more files for each part, but any option that appears in the last file loaded will overwrite previous settings. It is nevertheless useful if the previously loaded file contain options missing in the last loaded file.

It is worth noting that the `pgstar` inlist can be modified during run time, allowing an easier control of the plot seen[14]. The drawback is that a typo made while modifying the `pgstar` inlist will make the run crash. The other two inlists must be modified before (re-)starting the run, since they are read only at the beginning, but there is no need to recompile the code to make the changes operative: inlists are not a programming language, no logical expression can be used in them, but there is no need to compile them either.

3.3.1 More advanced settings

MESA offers to the user a large variety of settings, but they may not be enough. Even in this situation, the design of MESA allows modification of the code behavior without having to modify directly the source (which you may also be using for other parallel projects!). The trick is to use the `run_star_extras.f` in the folder `src` in your working directory. Once we copy the content of `run_star_extras.inc` in the `run_star_extras.f` in our working directory, we have in it functions to access each step of the computation of an evolutionary stage. Here we can calculate variables we would like to add to the `profile*.data` and/or the `history.data`, we can implement routines for a customized mass loss scheme, meshing algorithm, custom stopping and/or convergence criteria, etc... Note that any time the `run_star_extras.f` is modified, the code must be recompiled (while it is not needed when modifying the inlists). In fact, the `run_star_extras.f` file is a true FORTRAN file in which all the capabilities of the FORTRAN programming language can be used. For a large number of useful tutorials on the `run_star_extras.f`, see http://mesa.sourceforge.net/run_star_extras.html. We will propose an exercise on the use of `run_star_extras.f` in §4.0.4

3.4 How to start a run

To start a run for the first time, or after any modifications of the `run_star_extras.f` file, the code must be compiled: if the installation was done using the `MESA_SDK`, the compiler will already be installed, and in the default work directory there is a script to compile. The user just needs to go into the work directory and type `./mk`. Then, to start the run, the

[14] The user can either use the command `ctrl + Z` to pause the run, modify the `pgstar` inlist and then resume the run with the command `fg`; or just modify it in a separate bash shell.

appropriate script is `./rn`, and to restart from a photo (the binary files MESA saves) `./re <photo_name>`, which assumes the photo is saved in the folder `work/photos`. The work directory includes also a script to erase all the executable files, `./clean`, sometimes useful after a failed compilation.

4 Hands on session

We will try to perform a parameter and convergence study on each evolutionary stage of the evolution of a $10 M_{\odot}$ star, to explore the sensitivity of the lower threshold for massive stars to various parameters.

You can log in one of the two machines on which MESA is installed with either[15]:

```
ssh -YA mesawork@astr4pi.df.unipi.it ,
or
ssh -YA mesawork@astr18pi.df.unipi.it .
```

Then change directory with:

```
cd ./MESA_WORKSHOP ,
```

and you can work inside this folder.

The evolutionary stages we will try to explore are:

1. the main sequence;
2. to He ignition;
3. to He depletion;
4. (take home) to O central depletion;
5. (take home) Si burning;
6. (take home) To onset of core collapse.

At the end of each one we will save a model to be able to restart from it later. You will have to define the stopping criterion and tell it to MESA through your inlist.

Moreover, in your inlist you can change some parameters, the parameter space available is vast. Among many other things, I suggest trying to change some of the following and see how the structure changes:

- initial metallicity;
- $\alpha_{\rm mlt}$;
- nuclear reaction rates;
- spatial and temporal resolution (`varcontrol_target` and `mesh_delta_coeff`)

[15] The password will be provided during the workshop.

- nuclear network[16];
- rotation rate (if you really want to!);
- $\alpha_{\rm mlt}$, overshooting parameter;
- mass loss scheme and efficiency(η);
- ....

I suggest you make different models with different parameter settings of the first evolutionary stage and compare them. You can run more advanced evolutionary stages during the night.

4.0.1 Basic Guidelines

1. (log in and change directory to the one where you want to work:)

   ```
   cd /path/of/your/folder
   ```

2. copy a MESA working directory from `$MESA_DIR/star` in your current directory:

   ```
   cp -r $MESA_DIR/star/work ./
   ```

3. (rename your working directory to something meaningful and enter in it:)

   ```
   mv ./work ./somethingmeaningful && cd ./somethingmeaningful
   ```

4. copy from `$MESA_DIR/star` the `inlist_massive_defaults`, we will use it as a starting point and for the plots:

   ```
   cp $MESA_DIR/star/inlist_massive_defaults ./
   ```

5. edit[17] your main inlist to read the `inlist_massive_defaults` first and your `inlist_project` after. In this way we can overwrite some parameters of `inlist_massive_defaults` with our inlist. Your inlist should look like this at the end of this step:

   ```
   ! this is the master inlist that MESA reads when it starts.

   ! This file tells MESA to go look elsewhere for its configuration
   ! info. This makes changing between different inlists easier, by
   ! allowing you to easily change the name of the file that gets read.

   &star_job

       read_extra_star_job_inlist1 = .true.
   ```

[16] Do not pick a very large network if you want results in a reasonable run time, the computational cost increases roughly as $\sim N^2$ where N is the number of isotopes in the network. Moreover, during the initial phase those isotope will just sit there and do nothing, because T is too low for any nuclear processing of heavy elements.

[17] All steps involving editing of text files can be performed with any text editor

```
    extra_star_job_inlist1_name = 'inlist_massive_defaults'

    read_extra_star_job_inlist2 = .true.
    extra_star_job_inlist2_name = 'inlist_project'

/ ! end of star_job namelist


&controls

    read_extra_controls_inlist1 = .true.
    extra_controls_inlist1_name = 'inlist_massive_defaults'

    read_extra_controls_inlist2 = .true.
    extra_controls_inlist2_name = 'inlist_project'

/ ! end of controls namelist


&pgstar

    read_extra_pgstar_inlist1 = .true.
    extra_pgstar_inlist1_name = 'inlist_massive_defaults'

/ ! end of pgstar namelist
```

6. edit your `inlist_project` to have the settings you want. For this step you should search into the default files, read through the comments and find which setting you want to change and how to do it. You can either open these files and read them, or grep through them:

   ```
   grep -r ``something to search for'' $MESA_DIR/star/defaults
   ```

   The default files you want to look at are `controls.default` and `star_job.defaults`. Among the things that you want to change are (highlight to see the hints):

   - initial mass (hint:);
   - nuclear network (hint:);
   - initial metallicity and Y (hint:);
   - $\alpha_{\rm mlt}$ (hint:);
   - wind scheme and efficiency[18] (hint:);

[18] Not all wind scheme are appropriate for massive stars! I suggest the use of "Dutch"

- set a stopping condition for the TAMS (§2.5) (hint:

  );
- IMPORTANT: save a model when your run stops (and if you want a profile and a photo too), so that we can use it as a starting point (hint:

  );

7. check that everything works by compiling and then start the run[19]:

   `(./clean) && ./mk && ./rn`

8. to restart from a model you have to set *two* parameters in your `star_job` namelist.

At the end of each run you can compare your results. In computational physics it is common practice do make some plots (you can use whatever you want: python's matplotlib, gnuplot, etc...), since your numerical results usually cannot be expressed in analytic form.

4.0.2 PGstar plot tool

While you are running models, you can modify the plots you see on the screen, save them and post-process them into a movie (mp4). Detailed tutorials, including youtube videos, are available from http://mesastar.org/teaching-materials.

To try this MESA tool, you can either start a new run from the beginning and change your stopping criterion (or just comment what you set previously to use the default, which, for massive stars will be the onset of core collapse: $\max\{|v|\} \geq 1000\text{km s}^{-1}$), or go on with your run if it is going through a long lasting phase (i.e. post core He burning).
First of all you can add some more default plots by editing your main `inlist` to tell MESA to read also `inlist_pgstar` in the `pgstar` section. This can be done without stopping or pausing the run (you may want to open another terminal window), since the `pgstar` namelist is read at each iteration. Until now the main inlist should have been reading only `inlist_massive_defaults`. To see how add also `inlist_pgstar` scroll down. Beware of typos: they will make your entire run crash! Once the `pgstar` section of your main inlist looks like the following you should see an HR diagram and a $\log(T) \equiv \log(T)(\log(\rho))$ plot.

```
&pgstar

    read_extra_pgstar_inlist1 = .true.
    extra_pgstar_inlist1_name = 'inlist_massive_defaults'

    read_extra_pgstar_inlist2 = .true.
    extra_pgstar_inlist2_name = 'inlist_pgstar'

/ ! end of pgstar namelist
```

[19] The `./clean` command is an optional, useful after some compilation error

Then we can modify the plots by editing `inlist_pgstar` and/or `inlist_massive_defaults`. The relevant documentation is in `$MESA_DIR/star/defaults/pgstar.defaults`, which lists all the possible options and configurations. For example, you can try to make the HR diagram bigger, change the background color to white and save the a png file for each timestep. By default, MESA will create a subdirectory `png` inside your work directory and save the plots as pngs inside it. Once we have the png we can make a movie of them, showing the evolution. If MESA has been installed using the MESA SDK, it comes with a tool to post-process pngs into an mp4 movie. You can use it by running from the png subdirectory:

```
images_to_movie.sh ''file_name_root*.png'' movie_name.mp4
```

4.0.3 Effects of MLT++ on the evolution of Massive stars

This exercise follows the guidelines of one exercise of the MESA Summer School 2014 (see http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/quataert/monday-s-labs). The purpose is to explore the effects of the MLT++ on the evolution of different $M_{\rm ZAMS}$ massive stars.

1. Start a new work directory and rename it (see steps 2 and 3 in §4.0.1);
2. Download in your `src` directory the `run_star_extras.f` from http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/quataert/monday-s-labs. This file will just print on the standard output (usually the terminal window) more information;
3. Copy the `inlist_massive_WH07`[20] into your working directory and instruct the main inlist to read it;
4. Set a terminal condition, e.g. He depletion $Y_c \leq 0.01$;
5. Change the initial mass to something higher than $20 M_\odot$ and run it first with `okay_to_reduce_gradT = .true.` and `okay_to_reduce_gradT = .false.` (i.e. MLT++ on and off, respectively), and compare the results. Notice that very high initial masses could not make it to the end of your run.

4.0.4 The "other" hooks: custom wind mass loss scheme

Despite the fact MESA is an hydrostatic code, it cannot neglect the (dynamical) mass loss of stars: this influences the stellar evolution both at the surface (unveiling inner portions of the star, with different effective temperatures and chemical composition), and the core, through the change in the total mass. In order to simulate mass loss MESA, as most of the others stellar evolution codes, relies on time averaged recipes to extract mass from the surface. This are mostly based on observation or theoretical models, and suffer of very large uncertainties still unexplored in a systematic way. Many recipes are already built-in in MESA (see `controls.default` and `$MESA_DIR/star/private/winds.f`), but we can implement our own prescription using the `run_star_extras.f`. This exercise also gives the guideline to use

[20] The location of this file will be given during the workshop, since it is not known at the moment of writing

the "other hooks" allowing the user to bypass MESA routines and use their own instead. It will require a little coding in FORTRAN syntax.

We want to implement the wind scheme used to simulate massive stars in [6], which is the following:

- If $T_{\rm eff} < 15000K \Rightarrow$ Nieuwenhuijzen [7]:

$$\log(-\dot{M}) = -14.02 + 1.24\log(L/L_\odot) + 0.16\log(M/M_\odot) + 0.81\log(R/R_\odot) \ ; \quad (15)$$

- If $T_{\rm eff} > 15000K \Rightarrow$ Kudritzki [8]:

$$\dot{M} \equiv \dot{M}(\alpha,\delta,k,M,L) = \tilde{D}(\alpha,\delta,v(r=r_{\rm crit}),v_\infty)\left(\frac{\sigma_{\rm Th}kL}{4\pi c}\right)^{1/(\alpha-\delta)}\left(\frac{4\pi\alpha}{\sigma_{\rm Th}v_{th}}\right)^{\alpha/(\alpha-\delta)} \times \\ \times\left(\frac{1-\alpha}{GM(1-\Gamma}\right)^{(1-\alpha)/(\alpha-\delta)} \ ;$$

- If the surface hydrogen mass fraction $X_s < 0.4$ the star is considered a WR star $\Rightarrow$ Hamman [9–11]:

$$\log(-\dot{M}) = \begin{cases} -12.25 + 1.5\log\left(\frac{L}{L_\odot}\right) - 2.85X_s & \text{if } L > 4.5L_\odot \\ -35.8 + 6.8\log\left(\frac{L}{L_\odot}\right) & \text{if } L \le 4.5L_\odot \end{cases} . \quad (16)$$

Fortunately, MESA already includes the Nieuwenhuijzen and the Hamman prescriptions, so we just have to copy them into our `run_star_extras.f`.

The following is a step-by-step guide including some hints (as usual, highlight to see them). After each step you should go back into your working folder and try to compile, to make sure everything works.

1. First of all substitute the line

   ```
   include 'standard_run_star_extras.inc'
   ```

   with the content of the file it is pointing (hint:).

2. Look into `$MESA_DIR/star/other/other_wind.f`: it is a basic sample of routine to implement a wind. The variable `w` is just the $-\dot{M}$ we want to calculate. Copy the routine you find here into your `run_star_extras.f` and rename it to `WH07_wind` (hint:). Note that the declaration of the routine states which variables are known to the routine;

3. Go to the routine `extras_controls` in your `runs_star_extras.f` and tell MESA you want to use your routine (now called `WH07_wind` and in the `run_star_extras.f` file) for the wind instead of it's built in routine. To do so you have to use a pointer to your routine:

```
s% other_wind => WH07_wind
```

4. Define within the routine `WH_07` the variables you need to implement the scheme (hint:

   );

5. Add a line `contains` inside your function and copy the routines available in the MESA source files for Nieuwenhuijzen and Kudritzki schemes below it (hint:
). After this step your code should not compile: this is because the Kudritzki routine uses modules without any public interface. The work-around is to copy those modules in the source folder. (hint:

   );

6. Now go back above the `contains` line and initialize the variables you need to implement the wind. First of all you have to initialize a pointer to the data, which is done by calling the routine `get_star_ptr`. Just add this line after the variables declaration:

   ```
   call get_star_ptr(id,s,ierr)
   ```

   Since the mass is lost at the beginning of each timestep, we cannot access the variables through their names (they will be initialized later). (hint:

   );

7. Implement your wind scheme using if statements. MESA knows how to do base 10 logarithms and exponential functions: use `log10_cr(x)` and `exp10_cr(x)`, respectively.

8. Finally, tell your `inlist` you want to use your own routine instead of the default one. This can be done by setting the boolean `use_other_*`, in our case: `use_other_wind = .true.` If you grep for `use_other` in `$MESA_DIR/star/defaults/controls.defaults`, you can see which routine can easily be customized. If you ever need something else, just try to ask on the MESA mailing list (https://lists.sourceforge.net/lists/listinfo/MESA-users).

References

[1] B. Paxton, L. Bildsten, A. Dotter, F. Herwig, P. Lesaffre, and F. Timmes. *Astrophys. J. Supp. Ser.* , **192**, 3, 2011.

[2] B. Paxton, M. Cantiello, P. Arras, L. Bildsten, E. F. Brown, A. Dotter, C. Mankovich, M. H. Montgomery, D. Stello, F. X. Timmes, and R. Townsend. *Astrophys. J. Supp. Ser.* , **208**, 4, 2013.

[3] R. Kippenhahn, A. Weigert, and A. Weiss. *Stellar Structure and Evolution*. 2013.

[4] W.H. Press, S.A. Teukolsky, W. T. Vetterling, and B.P. Flannery. *Numerical Recipies - Third Edition*. Cambridge University Press, 2007.

[5] N. Smith. *Submitted to Ann. Rev. Astron. Astroph. , arXiv:1402.1237v1*, 2014.

[6] S. E. Woosley and A. Heger. *Phys. Rep.* , **442**, 269, 2007.

[7] H. Nieuwenhuijzen and C. de Jager. *Astron. Astrophys.*, **231**, 134, 1990.

[8] R. P. Kudritzki, A. Pauldrach, J. Puls, and D. C. Abbott. *Astron. Astrophys.*, **219**, 205, 1989.

[9] W.-R. Hamann, D. Schoenberner, and U. Heber. *Astron. Astrophys.*, **116**, 273, 1982.

[10] W.-R. Hamann, L. Koesterke, and U. Wessolowski. *Astron. Astrophys.*, **299**, 151, 1995.

[11] W. R. Hamann and L. Koesterke. *Astron. Astrophys.*, **335**, 1003, 1998.