MESA: a quick guide

Erika Korb

November 18, 2024

Contents

1 Installation

This guide focuses on the installation on Linux systems. For more details or other systems, go to the official page[1]. We will follow the installation described there and showed in the correspondent Youtube tutorial[2].

1.1 MESA SDK

Before installing MESA, it is recommended to get MESA SDK: a prebuilt set of compilers and run-time libraries that facilitate the MESA installation. We recommend to follow carefully the official guide on this[3]. We remind that the Ubuntu distribution has by default a Bourne shell, therefore in the "installation" section follow the correspondent instructions.

For instance, to install the SDK for the 22.05.1 version (at release) of MESA in the folder `~/MESA/22051` on Ubuntu do as follows:

- Download the file `mesasdk-x86_64-linux-22.6.1.tar.gz` and unzip it into the correct folder

```
$ tar xvfz mesasdk-x86\_64-linux-22.6.1.tar.gz -C ~/MESA/22051/
```

- Now we need to create the links to the newly created `mesasdk` folder. These links will be needed each time we open a shell to run MESA, therefore it is clever to create a function for it in the `~/.bashrc` file

```
function mesasdk-22051 {
export MESASDK_ROOT=~/MESA/22051/mesasdk
source $MESASDK_ROOT/bin/mesasdk_init.sh
echo "environment set for MESA SDK for MESA 22.05.1"
}
```

  Save the `~/.bashrc` file and simulate a new login session running in the home folder

```
$ source .bashrc
```

  And check that the newly create function works

```
$ mesasdk-22051
```

- Check that SDK is properly installed by checking that `gfortran` is correctly installed. Do so by running

```
$ gfortran --version
```

  and expect an output like this

```
GNU Fortran (GCC) 12.1.0
```

1.2 MESA

1.2.1 Download

Now let us download and install MESA. Note that the version of MESA referenced by MESA SDK can be a little bit old with respect to the newest available version of MESA.

For instance, in January 2023 the latest MESA SDK was still the one released with respect to the MESA 22.05.1 version, although the 22.11.1 version of MESA was already available. Therefore, we will unzip the correspondent file and have the main MESA directory in `~/MESA/22051/mesa-r221.1.1/`

[1] https://docs.mesastar.org/en/release-r22.11.1/installation.html
[2] https://www.youtube.com/watch?v=NmaLHFxpALg
[3] http://user.astro.wisc.edu/~townsend/static.php?ref=mesasdk

1.2.2 Set environment variables

Similarly to what we did with MESA SDK, we now update the `.bashrc` file with the variables needed to install MESA. We can create a single function that contains also the paths already set for mesasdk, so that we need to call only one function when we open a shell.

```
# set MESA_DIR to be the directory to which you downloaded MESA
# set OMP_NUM_THREADS to be the number of cores on your machine
# and then the paths to mesa sdk
function mesa-22111 {
export MESA_DIR=~/MESA/22051/mesa-r22.11.1
export OMP_NUM_THREADS=4
mesasdk-22051
echo "environment set for MESA 22.11.1"
}
```

Then apply changes to the `.bashrc` file

```
$ source ~/.bashrc
```

1.2.3 Installation

Finally, go inside the main MESA directory, execute the previously defined function to set the MESA environment and run the installation. Note that the MESA environment should be created every time we need to use MESA, thus, as a first action when we open a terminal for mesa, we need to run that command and it will work as far as we use that terminal.

```
$ cd mesa-r22.11.1
$ mesa-22111
$ ./install
```

Everything should run smooth. If it does not, look at the troubleshooting section in the official guide[4]

v. 7503 (MIST)

For the version used in Choi et al. 2016 to produce the MIST tracks - v. 7503 - you need to install `makedepf90` before calling the ./install command. The makedepf90 tar file is in the mesa/utils directory. Unpack it, go inside that directory and install it

```
$ tar -zxvf utils/makedepf90-2.8.8.tar.gz
$ cd makedepf90-2.8.8
$ ./configure
$ make all
$ sudo make install
```

In theory, everything works fine. However, it could happen that configuration does not work out for problems with C.

2 Running MESA

A detailed guide is available in the official site[5]. Nevertheless, here you can find a fast version of it.

2.1 Workflow

There is a general workflow to follow in order to run MESA.

- initialize paths and settings in a terminal window
- add physics and output details
- run the code via the *inlist* file

[4] https://docs.mesastar.org/en/release-r22.11.1/installation.html
[5] https://docs.mesastar.org/en/release-r22.11.1/using_mesa/running.html

2.1.1 Initialize

Go to the directory of your project and there open a terminal. First, create the MESA environment to ensure you have all path variables correctly set. Then copy the module folder you want to work with in the working directory you're in. Hereafter, we will adopt the /star/work folder that refers to the /star module. Once the folder is copied, we go inside that folder and compile the code.

```
$ mesa-22111
$ cp -r $MESA_DIR/star/work newfoldername
$ cd newfoldername
$ ./mk
```

2.1.2 Parameters and output setting

Details on the evolutionary physics and on output features are contained in three different places:

- *inlist_like* files in your working directory. Contains your main customizations.
- */src/run_star_extras* file, also in the working directory. Similar to *inlist* files, but here you can put many detailed conditions to leave the *inlist* files readable. For instance, in the *inlist* files you may adopt 3-4 lines to set wind details to 'Dutch' option, but leave to the *run_star_extras* file their detailed impementation. Also, in this file you may add conditions to print output files when given conditions are met.
- *$MESA_DIR/star/defaults/* files in the original MESA folder. Contains:
  - *history_columns.list*: parameter names and their definition. If un-commented, are the output quantities for the history.data file
  - *profile_columns*: parameter names, possible output columns for the profile files
  - *pgstar.defaults*: default settings for pgplots
  - *controls.defaults*: default settings for evolutionary processes (e.g. wind schemes). If the inlist does not specify an option, MESA applies the routines contained in this file. E.g. if the wind scheme is not specified in the inlist file and the *controls.defaults* file has empty strings in the hot_wind_scheme, cool_wind_RGB_scheme, and cool_wind_AGB_scheme options, no mass loss will be applied to the stellar models.

Profile columns As a rule of thumb, a good setting would make sure that the following quantities are flagged

- x,y,z (mass fractions, off by default)
- h1,he4,c12,n14,o16 (isotope abundances, off by default)

2.1.3 Run the *inlist* file

At this point, you have set everything and you can finally launch the MESA calculation with the command

```
    ./rn
```

This command, will execute the instructions inside the *inlist* file. Recall that the *inlist* file often is just an "index" of multiple *inlist-like* files.

2.1.4 Multiple runs, *inlist*, output folders and models

It could be very useful to evolve, for instance a star, "piece by piece". For example, one might want to

1. create a ZAMS model from the pre-main sequence star
2. evolve that ZAMS up to a given condition #1
3. evolve that star model up to a given condition #2
4. etc.

and save for each case

- *history.data* evolution
- profile evolution
- last profile
- last model

By default, MESA would save each evolution in the same LOGS and photos folders, overwriting each execution. Therefore, to achieve our goal we need to tell the inlist file to create different folders and save each "branch" in a given folder. Moreover, we will have to tell MESA to start each time the stellar evolution from the model calculated in each previous model. To do all of this, we will need a different inlist file with dedicated options for each branch. Then, we will call these "sub-inlists" one per time in the main inlist file and in sequential order the evolutions.

For example, first call the inlist linked to inlist1 to create the ZAMS model. Once the evolution is done and the ZAMS model is created, load it into inlist2 and launch again the main inlist but now link it only to the inlist2 (comment the link to inlist1). And proceed in this way.

2.2 Create a stellar model from ZAMS

WARNING: MESA ZAMS models exist only for Z=0.02 (See slide 54 of this guide[6]).

For instance, to create the model for a 58 $M_\odot$ star from the ZAMS to the TAMS, one could use the following *inlist_project*:

```
    &star_job

    ! create_pre_main_sequence_model = .false.

    ! to create a ZAMS not at Z=0.02, first relax to new metallicity
    relax_initial_Z = .true.
    new_Z = 0.0002

    ! save a model at the end of the run
    save_model_when_terminate = .true.
    save_model_filename = '58M_at_TAMS.mod'

    ! display on-screen plots
    pgstar_flag = .true.

    / !end of star_job namelist


    &controls

    ! starting specifications
    initial_mass = 58d0 ! in Msun units

    ! when to stop (TAMS)
    xa_central_lower_limit_species(1) = 'h1'
    xa_central_lower_limit(1) = 0.0001d0

    / ! end of controls namelist
```

What if we want to create a 58 $M_\odot$ model from scratch? In that case, we need to model our star from the pre-main sequence, thus, the indication for metallicity and the flag for the pre-main sequence model should change. Here you can see an example of code for this, taken from Renzo+2020

```
&star_job
create_pre_main_sequence_model = .true.
```

[6] https://fys.kuleuven.be/ster/education/sse/mesa-labs/mesalab-customizing-inlists-studentversion.pdf

```
show_log_description_at_start = .false.
save_model_when_terminate = .true.
! load_saved_model = .true.
save_model_filename = '58M_at_TAMS.mod'

! display on-screen plots
pgstar_flag = .true.

/ !end of star_job namelist


&controls

! starting specifications
initial_mass = 58d0 ! in Msun units

initial_Z = 0.0002d0
Zbase = 0.0002d0

! when to stop (TAMS)
xa_central_lower_limit_species(1) = 'h1'
xa_central_lower_limit(1) = 0.0001d0

/ ! end of controls namelist
```

3 Examples from literature

3.1 How to reproduce someone else's work: first steps to learn MESA

Usually on zenodo or here[7] you can find the copy of the /star/work directories adopted for published papers. These repositories and the tutorials of past MESA Schools are very useful resources to learn how to use MESA. A good idea to start is to try and reproduce some of such results. To do so, make sure you have installed the same MESA version of the paper or tutorial you are referring to. Then, create a working directory and inside it copy the /star/work directory of the authors. Go inside, open a terminal and set the correct MESA environment, for instance launching the `mesa-22111` command created in 1.2.2. Then, similarly to 2.1.1, execute the makefile. Only at this point you can run the simulation. Therefore, the three commands one should launch are

```
    $ mesa-22111      # or the one for the desired MESA environment
    $ ./mk
    $ ./rn
```

WARNING: It is important that you copy and make the *entire* original working directory. In fact, customized settings could be either directly stated in the "inlist"-like files but could also be hidden in the "/src/" files, especially in the "run_star_extras.f" one.

3.2 Literature works and MESA versions

Hereafter, we report a summary table of some useful works/examples in the literature that adopted MESA. We report the MESA version they used and the repository (if existent) from where download their scripts.

3.3 Renzo et al. 2020

Renzo et al. 2020[8] used MESA 12778 (1st March 2020) to generate two single stellar evolution models (58 and 42 $M_\odot$), merge them and evolve the post-merger product. Their repository is available here[9].

[7] https://cococubed.com/mesa_market/inlists.html
[8] https://ui.adsabs.harvard.edu/abs/2020ApJ...904L..13R/abstract
[9] https://zenodo.org/record/4062493#.ZAdcwdLMJhE

Work	MESA v.	Repository & Additional references
MIST	7503	https://github.com/jieunchoi/MIST_codes https://waps.cfa.harvard.edu/MIST/README_overview.pdf
Renzo et al. 2020	12778	https://zenodo.org/records/4062493 https://ui.adsabs.harvard.edu/abs/2020ApJ...904L..13R/abstract
Sabahit et al. 2023	12115	https://github.com/Apophis-1/VMS_Paper2/tree/main https://ui.adsabs.harvard.edu/abs/2023MNRAS.524.1529S/abstract
MESA Summer School 2023	23.05.1	https://mesahub.github.io/summer-school-2023/

Table 1: Summary of some useful repositories for works that used MESA

3.4 MIST tracks

4 Stellar winds

4.1 Pre-set options: Dutch winds

As explained here[10], MESA contains an already pre-set option, called "Dutch", to implement stellar winds according to the prescriptions provided by many Dutch authors (mainly: Vink et al. 2001, de Jager et al. 1988, Nugis & Lamers 2000). None of these prescriptions include, for now, the dependence on the Eddington factor described in Vink et al. 2011 and Graefner & Hafmann 2008.

4.2 Manual implementation

It is possible to manually set customized wind options by modifying the *run_star_extras.f* file in the /src folder in the working directory. Hereafter, for didactical purposes, we show an example of the implementation provided by Renzo et al. 2020 (already discussed in Sec. 3.3) and we will show how to fix some of their typos (our modifications will be shown in blue).

4.2.1 Wind theory

Hereafter we review the prescriptions implemented in this example and already collected in Renzo et al. 2017.

de Jager et al. 1988 They studied a sample of O to M stars in the Galaxy and derived a complex empirical relation for the mass loss. Usually, stellar codes implement the first order approximation of this relation (Eq. A.8 in Renzo et al. 2017):

$$\log_{10}(\dot{M}) = 1.769 \log_{10}(L/L_\odot) - 1.676 \log_{10}(T_{\rm eff}/K) - 8.158 \quad (1)$$

but it is possible to add a metallicity scaling $\dot{M} \propto (Z/Z_\odot)^{0.7}$ following the conclusions of the study conduced by Mauron & Josselin in 2011 on RSGs in the Milky Way and in the LMC.

Do RSG winds require a metallicity scaling?

Equation 1 implemented as it is in the *inlist* files use to produced simulations in Renzo et al. 2020 as well as the MIST tracks described in Choi et al. 2016. These prescriptions cause an important wind mass loss that abruptly reduces the stellar mass (even by 5 $M_\odot$) and is not in agreement with the properties (mass, radius, etc.) found in stars evolved with PARSEC up to the end of the core CO burning. It is not easy to decide which description is the correct one because RSG winds are still poorly understood. For instance, Choi et al. 2016 at page 10 of their paper recall the work of Mauron & Josselin 2011 that compared RSG wind prescriptions with RSG observations in the Milky Way and in the LMC. After this analysis, Mauron & Josselin concluded that the de Jager prescription is the better one available at the moment for RSG winds, but it is better to use it with a metallicity scaling. Technically speaking, the metallicity Z is the global one (as stated in the Mauron & Josselin work; *Zbase* in MESA) and not the surface one. In fact, we tested that adopting the surface metallicity in place of the original one we would quench too much the winds.

[10] https://docs.mesastar.org/en/release-r22.11.1/using_mesa/running.html?highlight=dutch#loading-a-model

```
subroutine eval_de_Jager_wind(w)
  ! de Jager, C., Nieuwenhuijzen, H., & van der Hucht, K. A. 1988, A&AS, 72, 259.
  real(dp), intent(out) :: w
  real(dp) :: log10w
  include 'formats'
  !log10w = 1.769d0*log10(L1/Lsun) - 1.676d0*log10(T1) - 8.158d0  ! original dJ88
  ! apply dJ88 correction suggested in Choi+2016 (p.10) and proposed by Mauron &
      Josselin 2011 from observed RSG
  log10w = 1.769d0*log10(L1/Lsun) - 1.676d0*log10(T1) - 8.158d0 + 0.7*log10((s% Zbase)
      /Zsolar)
  w = exp10(log10w)
end subroutine eval_de_Jager_wind
```

Vink et al. 2001 They ran numerical simulations for the winds in hot stars. They found that at $T_{\rm jump} \sim 25\,000\,\mathrm{K}$ there is the so-called "bi-stability jump": a region in the temperature space that discriminates two different wind regimes (see Fig. 1).

The bi-stability jump has a cool and a hot side, respectively for $T_{\rm eff} \lesssim 22\,500\,\mathrm{K}$ and $T_{\rm eff} \gtrsim 27\,500\,\mathrm{K}$. In the two sides, two different wind regimes are at work.

On the cool side, for $12\,500\,\mathrm{K} \leq T_{\rm eff} \leq 25\,000\,\mathrm{K}$, the mass loss in $M_\odot\ \mathrm{yr}^{-1}$ follows Eq. 25 in V01:

$$\begin{aligned}\log_{10}(\dot{M}) = &-6.688 \\ &+2.210\ \log_{10}\left(\frac{L/L_\odot}{10^5}\right) \\ &-1.339\ \log_{10}\left(\frac{M/M_\odot}{30}\right) \\ &-1.601\ \log_{10}\left(\frac{v_\infty/v_{\rm esc}}{2}\right) \\ &-1.07\ \log_{10}\left(\frac{T_{\rm eff}/\mathrm{K}}{20\ 000}\right) \\ &+0.85\ \log_{10}\left(\frac{Z}{Z_\odot}\right)\end{aligned} \tag{2}$$

On the hot side, for $27\,500\,\mathrm{K} \leq T_{\rm eff} \leq 50\,000\,\mathrm{K}$, the mass loss in $M_\odot\ \mathrm{yr}^{-1}$ follows Eq. 24 in V01:

$$\begin{aligned}\log_{10}(\dot{M}) = &-6.697 \\ &+2.194\ \log_{10}\left(\frac{L/L_\odot}{10^5}\right) \\ &-1.313\ \log_{10}\left(\frac{M/M_\odot}{30}\right) \\ &-1.226\ \log_{10}\left(\frac{v_\infty/v_{\rm esc}}{2}\right) \\ &+0.933\ \log_{10}\left(\frac{T_{\rm eff}/\mathrm{K}}{40\ 000}\right) \\ &-10.92\ \left[\log_{10}\left(\frac{T_{\rm eff}/\mathrm{K}}{40\ 000}\right)\right]^2 \\ &+0.85\ \log_{10}\left(\frac{Z}{Z_\odot}\right)\end{aligned} \tag{3}$$

Leitherer et al. in 1992 ([1]) found that the terminal wind velocity is proportional to the ion content present in the wind $v_\infty \propto Z^{0.13}$. In addition to this, V01 report that the Galactic ratio $v_\infty/v_{\rm esc}$ takes two different values on the two sides of the bi-stability jump. Therefore, in the two equations for the mass loss, this ratio can be rewritten as follows:

$$v_\infty/v_{\rm esc} = p \cdot Z^{0.13} \qquad p = \begin{cases} 2.6 & 27\,500\,{\rm K} \leq T_{\rm eff} \leq 50\,000\,{\rm K} \\ 1.30 & 12\,500\,{\rm K} \leq T_{\rm eff} \leq 22\,500\,{\rm K} \end{cases} \tag{4}$$

Inside the bi-stability jump, for $22\,500\,{\rm K} \leq T_{\rm eff} \leq 27\,500\,{\rm K}$, the mass loss must be interpolated. Renzo et al. 2020 suggest an interpolation with a *tanh* function, given the shape of the bi-stability jump shown in Fig. 1. They first calculate the mass loss at the extremes following the respective prescriptions

$$w_h = \dot{M}_{|T_{\rm eff=27\ 500K}} \tag{5}$$

$$w_c = \dot{M}_{|T_{\rm eff=22\ 500K}} \tag{6}$$

Then, they find the center of the bi-stability jump (in Kelvin) following Eq. 14 and 15 in V01

$$T_{\rm jump} = 10^3 \cdot [61.2 + 2.59 \cdot (-13.636 + 0.889 \cdot \log_{10}(Z/Z_\odot))] \tag{7}$$

and use it to perform the interpolation with the *tanh* function.

Renzo et al. 2020 *tanh* interpolation

Renzo et al. 2020 adopt the following interpolation

```
dT = 100d0
w = ((w_h - w_c)/2)*tanh((T1-Teff_jump)/(1d0*dT)) +(w_c+w_h)/2
```

In principle, the *tanh(x)* function reaches the asymptotic values of 1(-1) starting from $x \sim +(-)2\pi$. We are approximating, so we would like that the asymptotic values of the *tanh* function are reached in correspondence of $w_h = \dot{M}_{|T_{\rm eff=27\ 500K}}$ and $w_c = \dot{M}_{|T_{\rm eff=22\ 500K}}$. In other words, we want to enstablish a proportion between the argument x of the *tanh(x)* function and a coordinate defined as a function of the effective temperature of the star $f(T_{\rm eff} = T1)$:

$$x \in [-2\pi, 2\pi] \rightarrow \tanh(x) \in [-1, +1] \qquad \text{with} \quad \tanh(0) = 0 \tag{8}$$

$$T_{\rm eff} \in [27\ 500, 22\ 500] \rightarrow \tanh(f(T_{\rm eff})) \in [-1, +1] \qquad \text{with} \quad \tanh(T_{\rm eff} = T_{\rm jump}) = 0 \tag{9}$$

$$\tag{10}$$

From this equations emerges that we need to adopt a coordinate $T_x = -(T_{\rm eff} - T_{\rm jump})$, where the "-" arises from the fact that the maximum of the *tanh* function and of the bi-stability jump occurs in the cool side limit (see Fig. 1). We further need to re-scale this coordinate for the length of the x-axis, i.e.

$$T_x : \Delta T = x : 2\pi$$

In principle, the *tanh* function is symmetric with respect to the center but the position of $T_{\rm jump}$ is not symmetric with respect to the cool and hot extremes. Thus, we better define the semi-x-axis length in the temperature space assuming that the *tanh* is symmetric with respect to the side considered at the moment, namely we obtain

$$x = -(T_{\rm eff} - T_{\rm jump}) \quad \frac{2\pi}{\Delta T} \qquad \Delta T = \begin{cases} 27\ 500 - T_{\rm jump} & T_{\rm eff} \geq T_{\rm jump} \\ T_{\rm jump} - 22\ 500 & T_{\rm eff} < T_{\rm jump} \end{cases} \tag{11}$$

With this definition of x coordinate we obtain a value of $tanh(x) \in [-1, +1]$ as desired, thus this x is the coordinate function $f(T_{\rm eff})$ that we were looking for.

We need now to re-scale the $tanh(x) \in [-1, +1]$ output value to the actual mass loss values. Again, we'll build a proportion for the y-axis length

$$w_{\rm tanh} : \Delta w = tanh(x) : 1$$

The x-axis position of $T_{\rm jump}$ is not symmetric inside the considered interval, but its y-axis position is symmetric with respect to the maximum and minimum values of mass loss. Thus, we can define:

$$w_j = \dot{M}_{|T_{\rm eff=T_{jump}}} = \frac{w_h + w_c}{2} \qquad \Delta w = \frac{w_h - w_c}{2}$$

Eventually, the desired mass loss is obtained shifting and re-scaling the $tanh(x) \in [-1, +1]$ output on the y-axis

$$w_{\rm tanh} = w_j + \tanh(x) \cdot \Delta w \tag{12}$$

The implementation in MESA of this approach is provided below, outside this box. We point out that our implementation differs from the one of Renzo et al. 2020 for two reasons:

- they assume that the interval is symmetric with respect to $T_{\rm jump}$ also along the x-axis, in order to assume a fixed ΔT
- they assume that $\Delta T = 100\,{\rm K}$, without a clear explanation for the choice of the 100 K value

We provide now the full routine for wind calculation according to V01 as implemented in MESA. We recall that

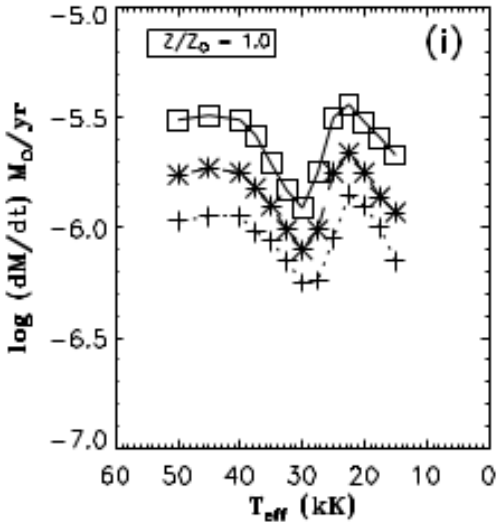

Figure 1: Example of bi-stability jump from Vink et al. 2001

the metallicity Z adopted by V01 is represented in MESA by the $Zfe = Zbase$, i.e. is referred to the initial metal content in the star.

```
subroutine eval_tanh_intrp_vink_wind(w)
  real(dp), intent(inout) :: w
  real(dp) :: alfa, w1, w2, w_c, w_h, w_j, dw, tx, Teff_jump, logMdot, dT,
      vinf_div_vesc
  w = 0

  if (T1 > 27500d0) then
     vinf_div_vesc = 2.6d0 ! this is the hot side galactic value
     vinf_div_vesc = vinf_div_vesc*pow(Zfe/Zsolar,0.13d0) ! corrected for Z
     logMdot = &
          - 6.697d0 &
          + 2.194d0*log10(L1/Lsun/1d5) &
          - 1.313d0*log10(M1/Msun/30) &
          - 1.226d0*log10(vinf_div_vesc/2d0) &
          + 0.933d0*log10(T1/4d4) &
          - 10.92d0*pow2(log10(T1/4d4)) &
          + 0.85d0*log10(Zfe/Zsolar)
     w1 = exp10(logMdot)
     w = w1
  else if (T1 < 22500d0) then
     vinf_div_vesc = 1.3d0 ! this is the cool side galactic value
     vinf_div_vesc = vinf_div_vesc*pow(Zfe/Zsolar,0.13d0) ! corrected for Z
     logMdot = &
          - 6.688d0 &
          + 2.210d0*log10(L1/Lsun/1d5) &
          - 1.339d0*log10(M1/Msun/30) &
          - 1.601d0*log10(vinf_div_vesc/2d0) &
          + 1.07d0*log10(T1/2d4) &
          + 0.85d0*log10(Zfe/Zsolar)
     w2 = exp10(logMdot)
     w = w2
  else ! interpolate inside the bi-stability jump
     !  first evaluate the mass loss rate one would have if T = 27500, all rest held
         constant
     vinf_div_vesc = 2.6d0 ! this is the hot side galactic value
```

```
      vinf_div_vesc = vinf_div_vesc*pow(Zfe/Zsolar,0.13d0) ! corrected for Z
      w_h =  &
           - 6.697d0 &
           + 2.194d0*log10(L1/Lsun/1d5) &
           - 1.313d0*log10(M1/Msun/30) &
           - 1.226d0*log10(vinf_div_vesc/2d0) &
           + 0.933d0*log10(27500/4d4) &
           - 10.92d0*pow2(log10(27500/4d4)) &
           + 0.85d0*log10(Zfe/Zsolar)
      w_h = exp10(w_h)
      !then evaluate the mass loss rate one would have if T= 22500, all rest held
         constant
      vinf_div_vesc = 1.3d0 ! this is the cool side galactic value
      vinf_div_vesc = vinf_div_vesc*pow(Zfe/Zsolar,0.13d0) ! corrected for Z
      w_c = &
           - 6.688d0 &
           + 2.210d0*log10(L1/Lsun/1d5) &
           - 1.339d0*log10(M1/Msun/30) &
           - 1.601d0*log10(vinf_div_vesc/2d0) &
           + 1.07d0*log10(22500/2d4) &
           + 0.85d0*log10(Zfe/Zsolar)
      w_c = exp10(w_c)
     !then interpolate between this two values
     ! use Vink et al 2001, eqns 14 and 15 to set "jump" temperature
     Teff_jump = 1d3*(61.2d0 + 2.59d0*(-13.636d0 + 0.889d0*log10(Zfe/Zsolar)))
     Tx = - (T1 - Teff_jump)  ! ''backward definition'': max of the jump occurs for
        Tmin
     if (Tx >= 0) then          ! assume asymmetric position of Teff_jump
       dT = 27500-Teff_jump     ! hot side
     else
       dT = Teff_jump - 22500   ! cool side
     ! print *, 'Teff_jump[K]', Teff_jump
     w_j = (w_c+w_h)/2     ! mass loss at Teff_jump
     dw = (w_h - w_c)/2    ! assume jump position symmetric on the y-axis
     w = w_j+ dw* tanh(Tx * 6.28/dT)
     end if
   end if
end subroutine eval_tanh_intrp_vink_wind
```

Nugis & Lamers 2000 We define a star as a Wolf-Rayet only where its surface hydrogen mass fraction drops below $X_H < 0.4$ and when its effective temperature is above the threshold *T_WR_full_on* defined in Sec. 4.2.2. According to Eq. 22 in Nugis & Lamers 2000, Wolf-Rayet winds can be described by the following empirical relation

$$\begin{aligned} \log_{10}(\dot{M}) = & -11.00 \\ & +1.29\ \log_{10}(L/L_\odot) \\ & +1.73\ \log_{10}(Y) \\ & +0.47\ \log_{10}(Z) \end{aligned} \tag{13}$$

where Y and Z are the helium and metal mass fractions, respectively. In MESA, this can be implemented as follows

```
    w = 1d-11 * pow(L1/Lsun,1.29d0) * pow(Y,1.7d0) * sqrt(s% Zbase)
```

Many authors in the literature proposed formulations for Wolf-Rayet winds, like Tramper et al. 2016. Therefore, we could build a script that toggles between our WR wind option, adopting a "switch variable" to be set in the *inlist* file. Here we call it *integer_ctrl(1)* and choose to set it equal to 1(2) to adopt the Nugis & Lamers (Tramper et al. 2016) option.

4.2.2 Full routine in MESA

Preliminary settings We'll define a routine called "Dutch_tanh_intrp_wind" that will return the wind value stored in the variable "w", initialized as "w=0".

```
subroutine  Dutch_tanh_intrp_wind(id, Lsurf, Msurf, Rsurf, Tsurf, X, Y, Z, w, ierr)
use chem_def
use star_def
integer, intent(in) :: id
real(dp), intent(in) :: Lsurf, Msurf, Rsurf, Tsurf, X, Y, Z ! surface values (cgs)
! NOTE: surface is outermost cell. not necessarily at photosphere.
! NOTE: don't assume that vars are set at this point.
! so if you want values other than those given as args,
! you should use values from s% xh(:,:) and s% xa(:,:) only.
! rather than things like s% Teff or s% lnT(:) which have not been set yet.
real(dp), intent(out) :: w ! wind in units of Msun/year (value is >= 0)
integer, intent(out) :: ierr
type (star_info), pointer :: s
real(dp) ::  L1, M1, R1, T1, xsurf, eta, Zsolar, Zfe, &
     log10w, w1, w2, T_high, T_low, alfa, fe, &
     T_WR_full_on, T_WR_full_off

integer :: h1, he4
real(dp) :: surface_h1, surface_he4

w = 0
ierr = 0
call get_star_ptr(id,s,ierr)
```

We then assign to new variables the quantities that we extract from the star, like surface temperature, luminosity or hydrogen mass fraction

```
L1 = Lsurf
M1 = Msurf
R1 = Rsurf
T1 = Tsurf

h1 = s% net_iso(ih1)
he4 = s% net_iso(ihe4)
if (h1 > 0) then
   surface_h1 = s% xa(h1,1)
else
   surface_h1 = 0
end if
if (he4 > 0) then
   surface_he4 = s% xa(he4,1)
else
   surface_he4 = 0
end if
xsurf = surface_h1    ! surface hydrogen mass fraction, used to switch on WR winds
ysurf = surface_he4
zsurf = 1 - (xsurf + zsurf)
```

Renzo et al. 2020 surface hydrogen mass fraction

In Renzo et al. 2020 the surface hydrogen mass fraction *xsurf* adopted to switch on the WR winds is set as $xsurf = X$ while here we chose $xsurf = surface\_h1$ as defined above. In fact, X is extracted from the outermost cell, but that cell is not necessary the photosphere. We extracted the *surface_h1* information through the *net_iso* command following the implementation suggested in 2016 by P. Marchant and available in its github repository (`https://github.com/orlox/mesa_input_data/blob/master/2016_binary_models/5_single/src/run_star_extras.f`).

We set as wind efficiency η the *Dutch_scaling_factor* appearing in the *inlist* file. By doing this, we'll apply the same efficiency to each of the following wind options.

```
eta = s% Dutch_scaling_factor
```

Renzo et al. 2020 Dutch scaling factor

In Renzo et al. 2020 the wind efficiency in the *inlist* file is set to be η =1.0. Instead, we set η =0.8 for convergence purposes of massive stars at $Z = 0.02$, even though the macro physical properties are not much affected by this change.

In the wind calculations we'll require two different metallicity values: *Zsolar* will indicate the fixed-a-priori solar metallicity, while *Zfe* metallicity will indicate the original *Zbase* metal content of the considered star. Remember that *Zbase* is not the Z variable: *Zbase* is the initial metallicity of the star, Z is the current one.

```
Zsolar = 0.019d0 ! for Vink et al formula, as adopted also in MIST (Choi+2016)
 Zfe = s% Zbase !! we don't want to use the actual Z polluted of CNO, but the baseline
    (see Tramper et al. 16)
```

Finally, we set temperature thresholds to distinguish different wind regimes. *T_low* and *T_high* will be used to discriminate between the de Jager et al. 1988 option and the Vink et al. 2001 option. Similarly, *T_WR_full_off* and *T_WR_full_on* will be used to switch between V01 and the Wolf-Rayet (WR) winds described in Nugis & Lamers 2000.

```
T_low = 10000d0            ! T < T_low applies only dJ88
T_high = 11000d0   ! lower limit for V01, extended as in MIST (p.10 of Choi+16)
                   ! lower limit for V01 (changed from Renzo+2020)
T_WR_full_off = 31622.8d0 ! log(T_WR_full_off)=4.5 ! start to mix V01 and NL00
T_WR_full_on = 50118.7d0  ! log(T_WR_full_on)=4.7  ! above, use only NL00
```

We were not able to find a reference for all of these settings. We point out that, unlike in Renzo et al. 2020, the lower limit of applicability of Vink et al. 2001 is extended from 12 500 K to 11 000 K according to the implementation adopted in Choi et al. 2016 to generate the MIST tracks (page 10 of Choi et al. 2016).

Wind regimes: interpolation The regime of validity of Vink et al. 2001 (V01) stands in between and partially includes the prescriptions of Nugis & Lamers 2000 (NL00) for Wolf-Rayet stars and the prescriptions of de Jager et al. 1988 (dJ88) for colder stars. Thus, we'll have to interpolate between the different regimes:

$$\dot{M} = \begin{cases} \text{NL00} & T_{\rm eff} \geq T_{\rm WR\ full\ on} \quad \text{and} \quad X_H < 0.4 \\ (1-\alpha_{\rm WR})\text{V01} + \alpha_{\rm WR}\text{NL00} & T_{\rm WR\ full\ off} \leq T_{\rm eff} \leq T_{\rm WR\ full\ on} \quad \text{and} \quad X_H < 0.4 \\ \text{V01} & T_{\rm eff} \geq T_{\rm high} \quad \text{and} \quad X_H \geq 0.4 \\ (1-\alpha)\text{dJ88} + \alpha\text{V01} & T_{\rm low} < T_{\rm eff} < T_{\rm high} \\ \text{dJ88} & T_{\rm eff} \leq T_{\rm low} \end{cases} \tag{14}$$

where

$$\alpha_{\rm WR} = \frac{T_{\rm eff} - T_{\rm WR\ full\ off}}{T_{\rm WR\ full\ on} - T_{\rm WR\ full\ off}} \tag{15}$$

implies that in the cold limit $T_{\rm eff} \to T_{\rm WR\ full\ off}$ we have $\alpha_{\rm WR} \to 0$: the wind prescription tends to the V01 limit. Similarly,

$$\alpha = \frac{T_{\rm eff} - T_{\rm low}}{T_{\rm high} - T_{\rm low}} \tag{16}$$

implies that in the cold limit $T_{\rm eff} \to T_{\rm low}$ we have $\alpha \to 0$: the wind prescription tends to the dJ88 limit.

Renzo et al. 2020 WR wind interpolation

Note that we exchanged the position of NL00 and V01 in the $T_{\rm WR\ full\ off} \leq T_{\rm eff} \leq T_{\rm WR\ full\ on}$ case with respect to the implementation of Renzo et al. 2020. In fact, they interpolated with the wrong $\alpha_{\rm WR}$ coefficients, writing

$$\dot{M} = (1 - \alpha_{\rm WR})\mathrm{NL00} + \alpha_{\rm WR}\mathrm{V01} \tag{17}$$

in place of

$$\dot{M} = (1 - \alpha_{\rm WR})\mathrm{V01} + \alpha_{\rm WR}\mathrm{NL00} \tag{18}$$

This means that they were wrongly enhancing the winds in the colder limit of this temperature range: they applied a Wolf-Rayet treatment in place of the softer V01 one.

Overall, the full interpolation is implemented as follows and recalls the subroutines previously defined in Sec. 4.2.1.

```
if ((xsurf < 0.4d0) .and. (T1 > T_WR_full_on)) then
    ! WR wind: surface H is low and T_surf > T_WR_full_on
    if(s%x_integer_ctrl(1)==1) then
       ! print *, 'using: N&L, eta=', eta
       w = 1d-11 * pow(L1/Lsun,1.29d0) * pow(Y,1.7d0) * sqrt(s% Zbase)
       w = w * eta
 else if ((xsurf <0.4d0) .and. ((T1>T_WR_full_off) .and. (T1< T_WR_full_on))) then
    ! interpolate Vink and WR wind for H-poor and T_WR_full_off <T_surf<T_WR_full_on
    ! print *, "interpolating Vink and WR wind"
    w1 = 1d-11 * pow(L1/Lsun,1.29d0) * pow(Y,1.7d0) * sqrt(s% Zbase)
    call eval_tanh_intrp_vink_wind(w2)
    alfa_WR = (T1 - T_WR_full_off)/(T_WR_full_on - T_WR_full_off)
    w = (1-alfa_WR)*w2 + alfa_WR*w1
    w = w * eta
 else
    ! H-rich, use pure vink
    if (T1 <= T_low) then
       ! print *, 'using: de_Jager, eta=', eta
       call eval_de_Jager_wind(w)
       w = w * eta
    else if (T1 >= T_high) then
       ! print *, 'using: Vink_tanh, eta=', eta
       call eval_tanh_intrp_vink_wind(w)
       w = w * eta
    else                ! transition
       ! print *, 'interpolating de Jager and Vink_tanh, eta=', eta
       call eval_de_Jager_wind(w1)
       call eval_tanh_intrp_vink_wind(w2)
       alfa = (T1 - T_low)/(T_high - T_low)
       w = (1-alfa)*w1 + alfa*w2
       w = w * eta
    end if
 end if
```

Informations on the pre-installed wind module can be found in $MESA_DIR/star/private/winds.f90

5 Mass transfer

5.1 Minilab of P. Marchant - MESA School 2022

Lab available here [11] and tuned for MESA v. 22.05.1.

[11] https://orlox.github.io/mesa2022_hmxb/