Gradu Amaierako Lana / Trabajo Fin de Grado
Fisika Gradua / Grado en Física

Modelling solar evolution with MESA and EVtwin

Egilea/Autor/a:
Alexander Olza Rodriguez

Zuzendaria/Director/a:
Raül Vera Jiménez

Leioa, 2020ko Ekainaren 18a /Leioa, 18 de Junio de 2020

Aknowledgements

I gratefully aknowledge the unconditional support of my parents and their profound belief in my abilities.

Many thanks to my partner Patricia and my friend Ada for nurturing me with encouragement and patience throughout the duration of this project.

I would like to extend my gratitude to my Director Raül for his constructive advice and practical suggestions.

Thanks also to the AMUSE GitHub community for helping me during the installation process.

Contents

Part I
Introduction and goals

1 About this work

Since the last century, many physicists and astronomers have devoted their careers to understanding the interior of stars.

Until recently, it has been difficult to achieve an experimental knowledge of the stellar structure hiding beneath the surface. Thus, scientists have resorted to models, suggesting a set of equations based on first principles (such as conservation of energy). The solution to this numerical problem constitutes a candidate to represent the internal structure of a star, that may change in time according to the proposed evolution equations.

In the case of the Sun, there is a vast amount of observational data that can be used to test whether the models are working. A valid solar model must make correct predictions for the present radius, luminosity, effective temperature and chemical composition at the surface. Moreover, since the 1970s, the field of helioseismology[1] has provided a new source of experimental information on the solar interior. This has driven the development of more sophisticated models.

In this Bachelor's Degree Thesis, we will first work towards the derivation of the equations of stellar structure and evolution, and then we will dive into the most widely used algorithm to solve them: The Henyey method.

After that, we will generate and evolve some solar models and study in detail some of their structural properties. To achieve this we will make use of AMUSE- the Astrophysical Multipurpose Software Environment developed by Simon Portegies Zwart and Steve McMillan.

2 About AMUSE

2.1 Motivation and internal functioning

There is plenty of independent codes for astrophysical simulation, each implemented uniquely and serving different purposes. As we will discuss in this section, this richness has some unfortunate downfalls.

[1]The study of the solar interior from the propagation of sound waves throughout the Sun.

Although many of them provide excellent solvers, most of the existing simulation codes are limited to a specific physical domain. However, astrophysical problems often spread through a wide ensemble of areas.

For example, to tackle the problem of a close binary system (or even a multiple system: see [1]), we need gravitational dynamics to follow the orbital movement and check for mass transfer episodes (which will occur if any of the stars overflow their Roche lobe [2]). Simultaneously, we need to monitor the stellar evolution of each component. Finally, in case of mass transfer, a smoothed particle hydrodynamics code will take over, and some new matter will fall on the accreting star. When mass transfer ends, the evolutionary forecast of each body will have changed dramatically, and the stellar evolution solver will need to be restarted with the new initial conditions.

But coupling all of these realms into a single simulation is not trivial: the codes need to exchange information as they run. The specificity of the data structures needed by each solver increases even further the difficulty for the users.

Thus, there was a clear need for a homogeneous, unifying framework. AMUSE meets these requirements. This project brings legacy codes closer to scientists of varying computing expertise, allowing them to perform simulations in a less cumbersome manner. Thanks to it, researchers can focus on the physics and restrict themselves to choosing the appropriate solvers and calling sequences without getting stuck in the many technical details.

The AMUSE environment works as a bridge between Python scripts and the so-called community codes, each in its native language (mostly Fortran, C/C++, and Java). This is done as follows:

- The *user script* calls the desired numerical solvers from Python, providing the necessary input in any system of units. This is the only user-written and problem-specific layer.

- The *manager* implements an object-oriented interface onto the communication layer, handling the unit conversion.

- The *communication layer* mediates between the manager and the community code layer. It transfers the information (such as initial conditions) from one to the other.

- The *community code layer* establishes interaction with community codes via MPI (the standard Message Passing Interface protocol, [2]) or SmartSockets [3]. The actual community codes live here too.

Nowadays, AMUSE includes a wide variety of codes for Gravitational Dynamics, Stellar Evolution, Hydrodynamics and Radiative Transfer. This enables users to solve numer-

[2]The region around a binary star where the material is gravitationally bound to that star. Any material outside this lobe may escape the system, fall onto the other star or orbit both stars.

ous problems- especially mixed problems that combine two or more of the main physical domains.

To learn more about the internal functioning of this environment and all of its functionalities, see [4]. In this work, however, we will restrict ourselves to writing Python scripts. We will make use of stellar evolution codes and describe them in subsequent sections, but we will regard them as modules available via AMUSE.

2.2 Brief tutorial : How to get started

First of all, you need to install the prerequisites and AMUSE itself, following the instructions at the AMUSE GitHub site[3].

You will see that AMUSE comes with a lot of examples. Trying to understand them is a good way to get started. The book [4] is a valuable resource, as it breaks down a lot of those scripts, but you can also begin by trying the tutorials at the documentation page[4]. Anyway, we will summarize the basics and work through a case study below.

2.2.1 The basics

First of all, we will define two spaces in which the variables in our programs can live:

We will name *local* space to the domain of our Python script, and *remote* space to that of the community codes. An essential feature of AMUSE is keeping these two domains separate.

We have stated that AMUSE provides an object-oriented interface. That means we will be declaring instances of objects (data-fields) with certain attributes, and they will be interacting with each other.

For example, a local object `SunEarth` may contain two other objects `Sun` and `Earth`. We can give each of them a set of properties such as `mass`, `position` and `velocity`. Let another object (of a different kind) represent the force of gravity: This will act on remote copies of `Sun` and `Earth` accessing and modifying their attributes.

Let us take a look at different AMUSE local data-structures:

- Quantity: A value (scalar or array) and a unit, appended with the vertical line operator.

```
    from amuse.units import units
    x=[1.0,0.0,0.0]|units.AU #a quantity
    dx=[0.0,1.0,0.0]|units.parsec
    print(x+dx) #prints [1.0, 206264.806245, 0.0] AU
    print x.value_in(units.parsec) #prints [4.85e-06,0,0], a value
```

Listing 1: Quantities

[3]https://amusecode.github.io/

[4]https://amuse.readthedocs.io/en/latest/tutorial/index.html

- Particle: A class of objects, usually with quantity attributes.
- Particle sets: 1D array of particles.

```
    from amuse.datamodel import Particle
    from amuse.units import units
    Sun=Particle(mass = 1 | units.MSun, radius=1 |units.RSun)
    Earth=Particle(mass = 1 | units.MEarth, radius=1 |units.REarth)
    SunEarth=Particles() #A particle set
    SunEarth.add_particle(Sun)
    SunEarth.add_particle(Earth)
    print SunEarth
    """           key           mass        radius
                    -           MSun          RSun
====================  ===========  ===========
13104127061724573785     1.000e+00    1.000e+00
 4579251171455465481     3.003e-06    9.160e-03
====================  ===========  ==========="""
```

Listing 2: Particles and particle sets

- Code instances: Bridges to a community code. Defining an object of this type creates a remote process in which the requested community code is running in the background, with its own ensemble of remote variables.

```
from amuse.lab import EVtwin
stellar_evolution=EVtwin()
```

Listing 3: Code instances

The characteristics of the remote variables depend on the implementation of the numerical solver we have called, but luckily we are not concerned with them. This is a fundamental advantage of AMUSE.

A typical workflow of an AMUSE program is shown in figure 1.

2.2.2 Case Study

Now we will solve a simple Gravitational Dynamics problem to become familiar with these concepts. The goal is to integrate the movement of the planets in our Solar System.

```
from amuse.units import units, nbody_system
from amuse.lab import ph4,new_solar_system
import numpy

def append_position(planet_positions,bodies,i):
  planet_positions.append([bodies[i].x.value_in(units.AU),bodies[i].y.
   value_in(units.AU),bodies[i].z.value_in(units.AU)])
```

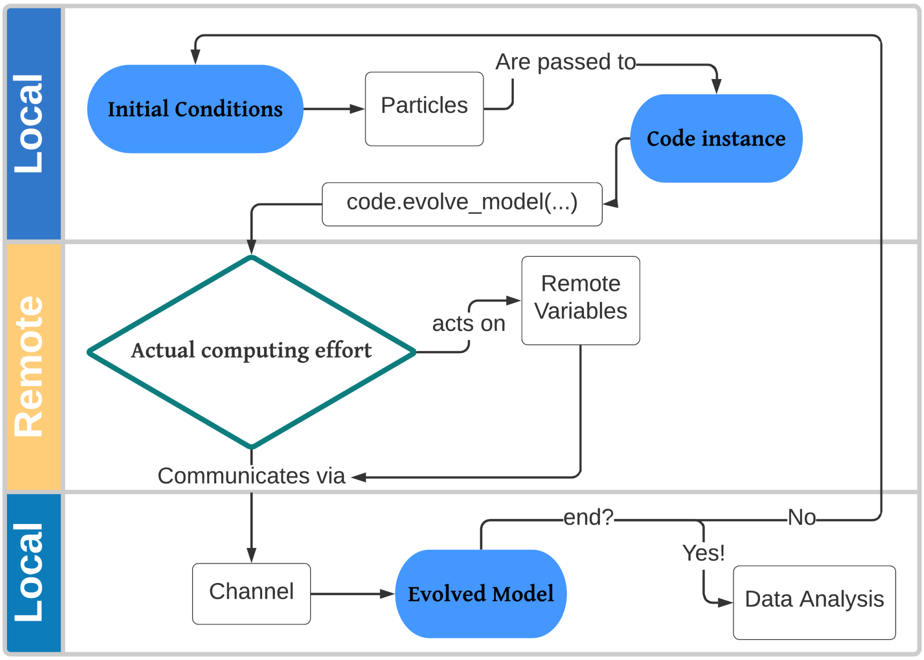

Figure 1: Flux diagram of an AMUSE program

```python
  return planet_positions

#MAIN STARTS HERE

bodies=new_solar_system()
```

Listing 4: amuse_tutorial.py

Now the variable `bodies` (which is a particle set) contains 9 particles with attributes ID key, name, mass, radius, position (vector attribute (x, y, z)) and velocity (v_x, v_y, v_z) We could have initialized the particles' mass, position and velocity manually, for example:

```python
 bodies=Particles(9)

 bodies.mass=[1,1.6e-7,2.4e-6,3e-6,3.2e-7,9.5e-4,2.9e-4,4.4e-5,5.1e-5]|
   units.MSun
```

The rest of the attributes are not necessary for our task.

If we type `print(bodies)` we get a table with the attributes of each body.

N-body codes work internally in N-body units, but our initial conditions are given in

SI units: We will ask AMUSE to make the conversion before passing the input to the N-body code.

The first step to do that is to define a converter between these two systems using the method `nbody_to_si` from the module `nbody_system`. It takes as input any pair of non-equivalent quantities (i.e. mass-length, luminosity-velocity...). The choice is arbitrary.

```
converter=nbody_system.nbody_to_si(1|units.MJupiter,1|units.AU)
```

ph4 is a N-body solver [5]. There are some others in AMUSE, but this is only a tutorial, so we are not concerned with choosing the best option. With the line below we initialize a "worker" of this code via MPI/Sockets, and tell AMUSE to use the converter before passing input to ph4. The worker is a completely separate process from this Python script, running independently. It will make a remote copy (gravity.particles) of all the variables passed as input, and will not modify the local ones (i.e. bodies). All communications (such as retrieving data) between ph4 and this script must be written explicitly.

```
gravity=ph4(converter)
gravity.particles.add_particles(bodies)
```

We have passed the initial conditions to ph4. The unit conversion has been done in the background. Gravity is not acting yet.

For validation, we will be checking energy conservation. The energy of the remote copy of bodies (initially identical to the local copy) is:

```
Energy_init=gravity.kinetic_energy+gravity.potential_energy
E=[Energy_init] #in nbody units
dE=[0]  # relative energy error (adimensional)

#We initialize time (in years)
t=0.0 |units.yr
dt=10.0|units.day
t_end=10.0 |units.yr

#We will only plot the orbits of these 4 bodies. The following will be
    arrays of VALUES, not quantities (explained below)
sun_position,mercury_position,venus_position,earth_position=[],[],[],[]
#We append the initial positions
sun_position=append_position(sun_position,bodies,0)
mercury_position=append_position(mercury_position,bodies,1)
venus_position=append_position(venus_position,bodies,2)
earth_position=append_position(earth_position,bodies,3)
```

Now, we define a communication channel between the remote copy, gravity.particles, and our local particle set, bodies. This is necessary to access the attributes of the particles in ph4, updating the positions and velocities of each body. However, it is not necessary to get the energy of the system as a whole (which we have already done).

In the while loop we let gravity act on the system for the desired amount of time, and we save the information at each timestep.

```
channel_to_bodies=gravity.particles.new_channel_to(bodies)

while t<t_end:
  gravity.evolve_model(t+dt) #This line makes ph4 integrate the
   equations of the nbody problem
  t+=dt
  channel_to_bodies.copy() #Now the variable bodies contains up-to-date
   information

  #Retrieving properties from ph4:
  E.append(gravity.kinetic_energy+gravity.potential_energy)
  dE.append((E[0]-E[-1])/E[0])
  #Saving positions:
  sun_position=append_position(sun_position,bodies,0)
  mercury_position=append_position(mercury_position,bodies,1)
  venus_position=append_position(venus_position,bodies,2)
  earth_position=append_position(earth_position,bodies,3)

gravity.stop()#This line terminates the ph4 process
```

Finally, we save and plot the trajectories and the energy error, and we get the figures 2 and 3. The functions to generate these figures are in `tutorial_plots.py` (Appendix A).

Observe that, each time `channel_to_bodies.copy()` is executed, we update the attributes of the object bodies. These are quantities. Notice also that, when calling `append_position (...)`, we have used the method `.value_in(units.AU)`. This transforms quantities to fixed numerical values. That is necessary for plotting, because Matplotlib (or any other plotting library) does not know what quantities are.

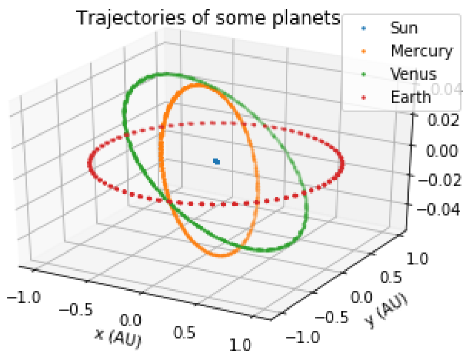

Figure 2: Orbits of some planets

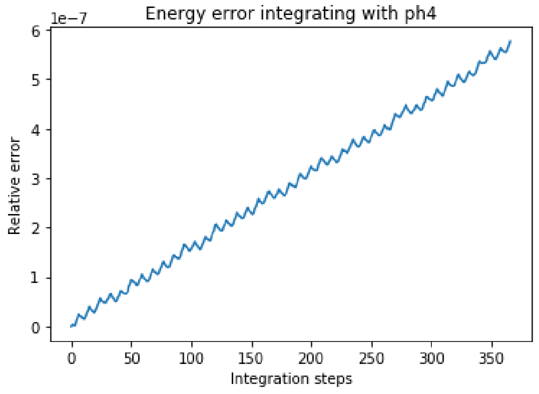

Figure 3: Energy loss of ph4 integration

Part II
Development

3 Obtaining the fundamental equations

We define the stellar structure as the set of the mass m, pressure P (or density ρ), luminosity L, temperature T and chemical abundances $X_i(i = 1, ..., I)$ for each concentric sphere of radius r from the centre to the surface.

Stellar structure and evolution are described, under the approximations described in the following section (spherical symmetry, amongst others), by the following equations:

$$
\begin{cases}
\dfrac{\partial r}{\partial m} = \dfrac{1}{4\pi r^2 \rho} & (1) \\[2ex]
\dfrac{\partial P}{\partial m} = -\dfrac{Gm}{4\pi r^4} & (2) \\[2ex]
\dfrac{\partial L}{\partial m} = \epsilon - c_P \dfrac{\partial T}{\partial t} + \dfrac{\delta}{\rho}\dfrac{\partial P}{\partial t} & (3) \\[2ex]
\dfrac{\partial T}{\partial m} = -\dfrac{GmT}{4\pi r^4 P} \nabla & (4) \\[2ex]
\dfrac{\partial X_i}{\partial t} = \dfrac{m_i}{\rho}\Big(\sum_j r_{ji} - \sum_k r_{ik}\Big) \qquad , \quad i = 1, ..., I & (5)
\end{cases}
$$

We will explain the variables on these equations soon. For the moment, we are only displaying the type of problem we face: A collection of $4 + I$ non-linear PDEs depending on m and t or, in the stationary case, 4 non-linear ODEs.

Below we will discuss the necessary approximations for these equations to be valid. Next, we will mention the key points to deduce them, without attempting to give a formal derivation.

3.1 Approximations in Stellar Evolution codes

Hereafter are a series of common assumptions used to simplify the vast majority of stellar evolution codes. They rely on the fact that equations (1)-(5) describe the macroscopic state of the star. Let us review the most remarkable:

1. **Spherical symmetry:** This reduces the problem from a set of PDEs in r, θ, φ, t to a set of PDEs in r, t, or a set of ODEs in the stationary case. In the way, we lose the possibility to describe the effects of rotation, oblateness and magnetic fields.

   Despite this being a regrettable loss the simplification level is huge, and it is a valid hypothesis for sun-like stars.

2. **Local Thermodynamic Equilibrium:**

   In stars, the temperature drops with increasing r. Then, for any r, there will be more photons moving towards the surface (coming from a hotter region) than towards the centre. This causes an energy flux that drives the luminosity throughout the star.

   LTE means that the temperature gradients pushing the energy transport are small. It is based on the mean free path of the photons being much smaller than the distance to the surface. Consequently, each photon interacts frequently with the stellar material.

3. **Treatment of Convection:**

   The energy transport in stars takes place via radiation, conduction and convection. This last phenomenon is impossible to treat accurately in a 1D code[5] because it contradicts the assumption of spherical symmetry. Most numerical solvers model it by the mixing length theory. According to this theory, each bubble of stellar material travels for a distance of l_m[6] before merging with its surroundings. l_m is a free parameter that can be determined empirically by fitting the stellar model to the observed properties of the Sun.

4. **Equation of state:**

   Different regions inside the star require varying contributions of the ideal gas, radiation and degenerate pressure, which can also change during the star's lifetime. The choice of tables for the equation of state is another characteristic of each numerical solver.

5. **Opacity:**

   In the passage about LTE, we have stated that the mean free path l_p of photons is much shorter than the distance to the surface. Opacity $\kappa = \frac{1}{\rho l_p}$ describes the difficulties that light faces in its way to the surface.

   There are various opacity sources in stars. For example, absorption of a photon promoting an electron in an atom to a higher energy level (*bound-bound* transition), ionizing absorption (*bound-free*) or Compton scattering (*free-free*).

[5]A code assuming spherical symmetry, thus, treating a one-dimensional spatial problem.

[6]l_m is related to the mixing length ratio $\alpha = l_m/H_p$ where $H_p = -P\frac{dr}{dP}$ is the so-called *pressure scale height*. In stellar structure and evolution, a common value is $\alpha = 2$.

These contributions depend on frequency, but it is common to consider κ as an average of ν. The most customary is the Rosseland mean, giving greater weight to high frequencies. Elaboration of opacity tables for different chemical compositions is an arduous numerical procedure, taking into account the statistics and quantum mechanics of stellar material [6]. The choice of tables is a determinant feature of each code.

3.2 Justifying the equations

Now we will go through equations (1)-(5), explaining their origin. There is a full derivation of all of them in [7] and in [8]. Here we will summarize the former.

3.2.1 Coordinate choice

There are two usual ways of describing a spherically symmetric star: Euler or Lagrange coordinates.

Euler's take the distance to the centre, r, as the independent variable. Then $m(r)$ satisfies $\frac{\partial m}{\partial r} = 4\pi r^2 \rho$. Lagrangian coordinates use m as an independent variable. Thus,

$$\frac{\partial r}{\partial m} = \frac{1}{4\pi r^2 \rho} \quad . \tag{6}$$

3.2.2 Hydrostatic Equilibrium

For most of its lifetime we can contemplate that stellar material is not accelerated: Pressure and gravity compensate for each mass element. The star evolves through a series of states in hydrostatic equilibrium.

Considering a spherical shell of thickness dr at a distance r from the origin, its mass per unit surface is ρdr and gravity exerts a force $f_g = -\frac{Gm}{r^2}\rho dr$ per unit area. At the same time, the outward pressure is a monotonically decreasing function of r. Thus, there will be a pressure gradient inside our infinitesimal shell: $P_i - P_e = -\frac{\partial P}{\partial r} dr > 0$.

In hydrostatic equilibrium, $\frac{\partial P}{\partial r} = f_g$. In Lagrangian coordinates, making the change of variable $\frac{\partial r}{\partial m} = (4\pi r^2 \rho)^{-1}$, we obtain equation 7:

$$\frac{\partial P}{\partial m} = -\frac{Gm}{4\pi r^4} \quad . \tag{7}$$

3.2.3 Conservation of energy

The luminosity $L(r)$ is the energy per unit time emitted by each spherical shell. It is zero at the center, but contrary to other monotonous functions it can have a complicated form because of the distribution of energy sources.

In each shell of thickness dm there may be an increment dL due to nuclear reactions, temperature changes or expansion/contraction.

In the stationary case the only term is the nuclear contribution, so $dL = \epsilon dm$ and $\frac{\partial L}{\partial m} = \epsilon$, where ϵ is the nuclear energy per unit mass and unit time. It depends on T, ρ and the chemical composition. If a $p \to q$ reaction releases an energy e_{pq}, its contribution to ϵ will be proportional to this energy and to the reaction rate (r_{pq}, dependent of temperature and p's local abundance).

In the non-stationary case, we have the two additional terms in the equation for energy conservation. They arise from thermodynamic relations. The shell may change its internal energy or its thickness, exchanging work (PdV) with the adjacent layers. Then, the extra heat added to the shell in an interval dt is $dq = (\epsilon - \frac{\partial L}{\partial m})dt$.

As shown in Chapter 4 of [7], the First Law of thermodynamics can be expressed as $dq = c_P dT - \frac{\delta}{\rho} dP$, where $\delta = -\left(\frac{\partial \ln \rho}{\partial \ln T}\right)_P$ and $c_P = \left(\frac{dq}{dT}\right)_P$. Using this we get rid of dq:

$$\frac{dq}{dt} = c_P \frac{\partial T}{\partial t} - \frac{\delta}{\rho}\frac{\partial P}{\partial t} \to \frac{\partial L}{\partial m} = \epsilon - c_P \frac{\partial T}{\partial t} - \frac{\delta}{\rho}\frac{\partial P}{\partial t} \quad . \tag{8}$$

3.2.4 Energy transport

Energy transport in stars happens by radiation, conduction and convection, depending of the local features of each zone. These three mechanisms are fed by a temperature gradient that allows the exchange of “particles” (photons, electrons or bubbles of stellar material) between layers of different temperature. Now we will sketch how to obtain the energy transport equation depending of the dominant transport mode.

- **Radiation:**

  The mean free path l_p of a photon in the stellar interior is of the order of centimeters [7]. Thus, radiation can be considered as a diffusion problem, with an energy flux[7] $\boldsymbol{F} = -\frac{1}{3}cl_p\boldsymbol{\nabla}U$ where c is the speed of light in vacuum and U the internal energy, with $U = aT^4 \rightarrow \frac{\partial U}{\partial r} = 4aT^3\frac{\partial T}{\partial r}$. Substituting $l_p = 1/(\kappa\rho)$, we obtain a heat equation:

$$F = -\frac{4ac}{3}\frac{T^3}{\kappa\rho}\frac{\partial T}{\partial r}$$

  Taking into account that $F = L/4\pi r^2$, the radiative temperature gradient is

$$\frac{\partial T}{\partial r} = -\frac{3}{16\pi ac}\frac{\kappa\rho L}{r^2T^3} \quad .$$

  Finally, with the appropriate change of variable, we write this in Lagrangian coordinates:

$$\frac{\partial T}{\partial m} = -\frac{3}{64\pi^2 ac}\frac{\kappa L}{r^4T^3} \quad . \tag{9}$$

  The diffussion approximation is no longer valid in the outermost layers, where the density is low and l_p grows.

- **Conduction:**

  This energy transport mode is based on collisions between particles with random thermal motion. The analogy with radiation is complete, but the diffusion coefficient is much smaller. Therefore, equation 9 is valid for both processes when a conduction term is added to the opacity. With the redefinition $1/\kappa = 1/\kappa_{rad} + 1/\kappa_{cd}$, the previous equations account for both mechanisms.

  Let us write (9) more conveniently. Dividing by equation 7,

$$\left(\frac{dT}{dP}\right)_{rad} = \frac{\partial T/\partial m}{\partial P/\partial m} = \frac{3}{16\pi acG}\frac{\kappa L}{mT^3} \quad .$$

  Defining $\nabla_{rad} = (d\ln T/d\ln P)_{rad}$,

[7]We use bold symbols for vectors.

$$\nabla_{rad} = \frac{3}{16\pi acG}\frac{\kappa LP}{mT^4} \quad .$$

∇_{rad} describes variation of T with depth, because P monotonically decreasing and can be interpreted as a parametrization of r. Substituting $\nabla = \nabla_{rad}$ we obtain the following equation for the radiative and conductive cases:

$$\frac{\partial T}{\partial m} = -\frac{GmT}{4\pi r^4 P}\nabla \tag{10}$$

- **Convection:**

  If the buoyancy forces are sufficient, energy may be transported by the exchange of macroscopic blobs of stellar material. Instead of solving the underlying Navier-Stokes equations, stellar evolution computations describe convection by the mixing-length theory. If this is the dominant transport mechanism, equation 10 will have a different ∇. When working out the mixing-length theory, one finds that ∇_{conv} is the solution of a cubic equation that must be obtained numerically.

  For chemically homogeneous regions, Schwarzschild's criterion determines the prevailing transport mechanism:

$$\begin{cases} \nabla = \nabla_{rad} & \text{if} \quad \nabla_{rad} < \nabla_{ad} \\ \nabla = \nabla_{conv} & \text{if} \quad \nabla_{rad} > \nabla_{ad} \end{cases} \quad ; \tag{11}$$

  where $\nabla_{ad} = \left(\frac{\partial \ln T}{\partial \ln P}\right)$ is the thermal variation of a mass element contracting adiabatically. This criterion is evaluated for each shell in the star using the local values of P, T, ρ. Depending on the result, we use equation 10 with $\nabla = \nabla_{rad}$ or we need to look for ∇_{conv}.

3.2.5 Chemical composition

We denote by $X_i(m,t)$ the fraction of type i nuclei in a given region at a given moment. Then, $\sum_i X_i = 1$, and X_i is related to the particle number density by $X_i = \frac{m_i n_i}{\rho}$. In radiative regions, there is no mixing between the composition of different layers, and the only source of changes in X_i are the nuclear reactions. Let r_{lm} be the total rate of reactions that turn nuclei of type l into nuclei of type m. The variation in n_i will be:

$$\frac{\partial n_i}{\partial t} = \sum_j r_{ji} - \sum_k r_{ik} \Rightarrow \frac{\partial X_i}{\partial t} = \frac{m_i}{\rho}\Big(\sum_j r_{ji} - \sum_k r_{ik}\Big) \quad . \tag{12}$$

We have obtained a set of I differential equations, called a "nuclear reaction network".

In convective regions, the mixing caused by the moving blobs is much faster than the nuclear timescale. Hence, we can consider that such zones have a uniform composition.

4 Numerical Methods

Now we will dive into the method that solves equations (1)-(5). There are different versions available in the literature, including [7],[9] and the original [10]. The first two are very similar. Here we will follow the one in [9].

4.1 Boundary conditions

Let us consider the star as a set of N grid points. Defining $(r, P, L, T) \equiv (x_1, x_2, x_3, x_4) = \boldsymbol{x}$, we want to obtain $\boldsymbol{x}^n$ for $n = 1, ..., N$. A subscript indicates the variable we are referring to, and a superscript indicates the grid point.

The mesh points must be distributed in a way that none of the variables changes by more than a few per cent between adjacent cells. Besides, the cell size must vary slowly to avoid numerical abnormalities. The number of grid points can range from a couple hundred to a couple thousand. In the simulations that we will describe in Section 3, the code EVtwin used 199 zones in all cases. With composition AGS[11], MESA used 604 zones at ZAMS and 862 at the First Giant Branch. During the Helium Flash observed with our alternative composition, MESA used 1021 cells.

At the center of the star we have $m, r, L = 0$ and P_c, T_c to be determined. To avoid the central singularities in equations (1)-(5), we expand each of them in a Taylor series around the origin. We will call these central equations $G^1_\alpha, \quad \alpha = 1, ..., 4$, encoded as the vector $\boldsymbol{G^1}$:

$$\boldsymbol{G^1} = \begin{cases} r - \left(\frac{3}{4\pi\rho_c}\right)^{1/3} m^{1/3} = 0 \\ P - P_c + \frac{3G}{4\pi}\left(\frac{4\pi\rho_c}{3}\right)^{4/3} m^{2/3} = 0 \\ L - \epsilon_c m = 0 \\ T - T_c + \frac{G}{2}\left(\frac{4\pi}{3}\right) T_c m^{2/3} \nabla_c = 0 \end{cases} \tag{13}$$

The boundary conditions at the surface will be obtained by the chosen model for the stellar atmosphere, which is beyond the scope of this work. In any case, we will have 4 functions of the following kind, where $\Delta r^{atm}, P^{atm}, T^{atm}$ are the results of the atmospheric calculation (and thus data in our setting):

$$\boldsymbol{G^N} = \begin{cases} r^N - r^{N-1} - \Delta r^{atm} = 0 \\ P^N - P^{atm} = 0 \\ L^N - L^{N-1} = 0 \\ T^N - T^{atm} = 0 \end{cases} \tag{14}$$

$L^N - L^{N-1} = 0$ expresses the fact that there is no energy generation in the atmosphere.

In the stellar interior we will apply finite difference versions of equations (1)-(4), now expressed as $G^n_\alpha = 0$ with $n = 2, ..., N-1$ and $\alpha = 1, ..., 4$ and encoded as $\boldsymbol{G^n}$. Equation (5) will also be discretized. These equations are valid at a point below the surface, deep enough that the diffusion approximation still holds: If we are too close to the surface, the mean free path of the photons is comparable to the distance to the exterior and equation 10 is no longer correct.

4.2 The Henyey method

Equations (1-4) are solved implicitly and simultaneously with the boundary conditions we have just described. Additional relations or tables will be used to obtain ρ, ϵ, κ... as a function of the local P, T, X_i. The structure at the time $t + \Delta t$ is calculated from a previous model at t, and the first model may be obtained by a shooting method[8].

Each equation only depends on r, P, L, T in two of the three neighbouring cells: $G^n_\alpha(\boldsymbol{x}^m, m = n, n \pm 1) = 0$. Expanding G in a Taylor series to first order,

$$G^n_\alpha + \sum_{\beta=1}^{4} \frac{\partial G^n_\alpha}{\partial x^{n-1}_\beta} \delta x^{n-1}_\beta + \sum_{\beta=1}^{4} \frac{\partial G^n_\alpha}{\partial x^{n}_\beta} \delta x^{n}_\beta + \sum_{\beta=1}^{4} \frac{\partial G^n_\alpha}{\partial x^{n+i}_\beta} \delta x^{n+1}_\beta = 0 \tag{15}$$

From an initial guess for the x^n_α, we solve the linearized system (15) for the corrections δx^n_α, add them to the variables and iterate until $\boldsymbol{G^n}, \boldsymbol{\delta x^n} \to 0$. The system can be

[8]A common technique to solve split boundary-value problems. In this case it consists in integrating layer by layer with Runge-Kutta or an equivalent integrator. One needs to make an initial guess for P_c, T_c, and discard the solutions that do not meet the outer boundary conditions. This is repeated with different P_c, T_c until the outer boundary conditions are satisfied.

expressed in sparse-matrix form. Let us define some 4×4 matrices:

$$\begin{cases} C^n_{\alpha\beta} = \frac{\partial G^n_\alpha}{\partial x^{n-1}_\beta}; \quad C^1_{\alpha\beta} = 0 \\ D^n_{\alpha\beta} = \frac{\partial G^n_\alpha}{\partial x^n_\beta} \\ E^n_{\alpha\beta} = \frac{\partial G^n_\alpha}{\partial x^{n+1}_\beta}; \quad E^N_{\alpha\beta} = 0 \end{cases} \tag{16}$$

Then the so-called Henyey system is

$$\begin{bmatrix} D^1 & E^1 & 0 & 0 & & & 0 \\ C^2 & D^2 & E^2 & 0 & & & 0 \\ 0 & C^3 & D^3 & E^3 & 0 & & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0\cdot & \cdot & \cdot & 0 & C^{N-1} & D^{N-1} & E^{N-1} \\ 0\cdot & \cdot & \cdot & 0 & 0 & C^{N-1} & D^{N-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta x^1} \\ \boldsymbol{\delta x^2} \\ \boldsymbol{\delta x^3} \\ \cdot \\ \cdot \\ \boldsymbol{\delta x^{N-1}} \\ \boldsymbol{\delta x^N} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{G^1} \\ \boldsymbol{G^2} \\ \boldsymbol{G^3} \\ \cdot \\ \cdot \\ \boldsymbol{G^{N-1}} \\ \boldsymbol{G^N} \end{bmatrix} \tag{17}$$

This sparse matrix of dimension $4N$ is block-tridiagonal and has a non zero determinant[7]. We will call it the Henyey matrix, H. To solve for the corrections we must find the inverse of H. Because of the special form of the matrix, there is a specific way of doing this, based on elimination and back substitution. It is described in [9]. Instead of computing H^{-1} by brute force, this method inverts a set of 4×4 matrices, built from C^n, D^n, E^n. The algorithm is described in [9].

Figure 4 summarizes the process to compute a time series of stellar models. We start from an initial guess for $\boldsymbol{x^n}(t)$, with an appropriate grid.

First, we verify the Schwarzschild criterion in each grid point. To do so, we need to compute ∇_{rad} and ∇_{ad} with the profiles in the initial guess. If convection dominates in any stellar region, we compute ∇_{conv}.

Once we know which energy transport equation to use, we test whether $G^n_\alpha \leq e$ where e is a small tolerance value. If the initial guess fails to meet this condition, we compute H and solve the Henyey system, adding $\boldsymbol{\delta x}$ to r, P, L, T until the corrections are small enough or $G^n_\alpha \leq e$.

Then, we have obtained a valid stellar model for t. We are ready to evolve to the next timestep. To do that, we solve the discretized version of the nuclear reaction network (equation 5). This involves computing the reaction rates, which depend on r, P, L, T for t. The initial guess of r, P, L, T for $t+\Delta t$ can be the current stellar model or an extrapolation based on $t - \Delta t$ and t.

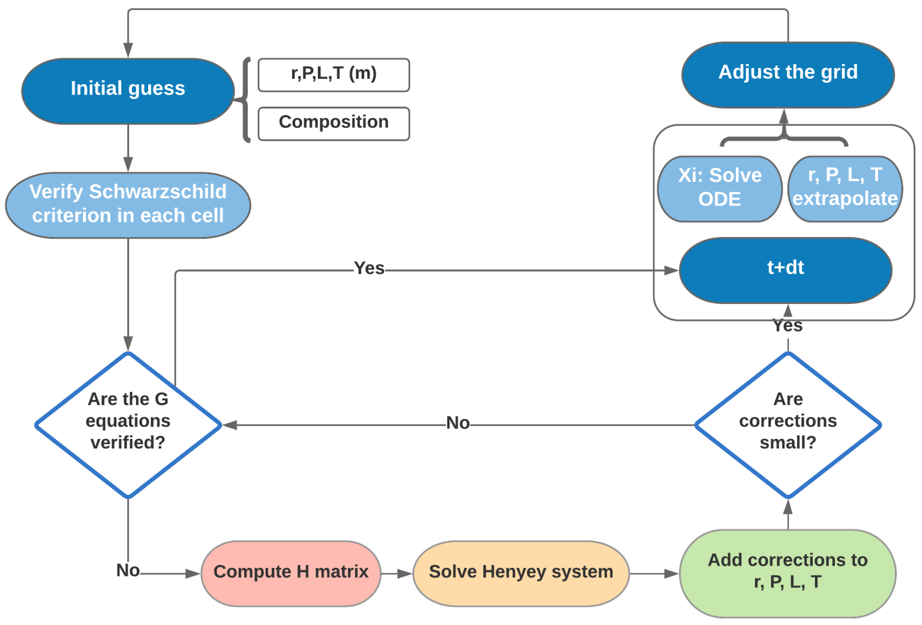

Figure 4: Procedure of a stellar evolution calculation

4.2.1 MESA

Modules for Experiments in Stellar Astrophysics (MESA) [12] is a collection of open-source software that includes a 1D stellar evolution code, `MESA star`, hereafter MESA. This is a Henyey solver that relies on a variety of complementary modules: `atm` for atmospheres, `eos` for equations of state, `rates` for nuclear reaction rates, `kap` for opacities, `mlt` for mixing lenght theory, `mtx` for linear algebra matrix solvers, amongst others.

For the equations of state, MESA uses the 2005 version of tables [13]. At lower temperatures and densities, it uses [14] tables. `eos` provides tables for $X = (0.0, 0.2, 0.4, 0.6, 0.8, 1.0)$ and $Z = (0.0, 0.02, 0.04)$, and uses interpolation routines for other mixtures.

The module `rates` uses NACRE [15] reaction rates when available, and [16] reaction rates otherwise.

`kap` constructs the MESA opacity tables by combining different sources depending on the values of ρ, T. These references include [17] and [18] for electron conduction opacities and [19] and OPAL [13] for radiative opacities. The combined tables are available for a set of different compositions, and they are interpolated for other cases. The derivatives needed to compute the Henyey matrix are implemented analytically, and the mesh refinement is automatic.

Now we will describe how MESA works to obtain a time series of stellar models. The starting point can be a saved model from a previous run or one of the precomputed ZAMS[9] models distributed with MESA. MESA checks the structure profiles at the beginning of each timestep and adjusts the mesh as necessary. The remeshing method is designed to minimize the number of cells while ensuring that the relative changes in r, P, L, T are lower than a given limit. This optimizes execution time because it reduces the number of matrix inversions. The algorithm increases resolution in regions of intense nuclear burning, lowering the threshold for allowed change between cells.

After finding an appropriate grid, the Henyey system is solved using matrix inversion algorithms from the module `mtx` until the corrections meet convergence criteria. This usually takes 2 or 3 iterations in regular stellar stages. During challenging phases such as the He Flash, convergence criteria may be automatically adjusted.

Once the solver has found a valid model, it generates output files and progresses to the next step. The time interval between consecutive stellar models must be large enough to enable efficient evolution but small enough to allow convergence. MESA monitors the advances keeping all the relative changes under a certain threshold. If the code cannot achieve convergence, the timestep is automatically reduced. It the problem persists, MESA returns to the previous model and proceeds with a smaller timestep. This continues until the model converges or the timestep reaches a pre-defined minimum.

4.2.2 EVtwin

EVtwin is another implementation of the Henyey method. It can compute both single and binary evolution. It is described in [20]. Contrary to MESA, the number of cells in EVtwin is fixed (199). The mesh is non-Lagrangian[10], and the cells are redistributed according to the computational needs.

The grid construction method is very different. It is formulated in detail in [21] as a variational problem. Instead of checking the r, P, L, T profiles and constructing the mesh at the beginning of each timestep, EVtwin solves for the mesh point distribution, the structure and the composition in a single iterative step. The program tracks the content of $^1H,^4He,^{12}C$ and ^{16}O using [16] reaction rates. Therefore, it solves a set of 9 differential equations (4 for the structure, 4 for the composition and 1 for the grid). The structural variables are $\log r, \log P, L, \log T$ rather than r, P, L, T.

[9]Zero Age Main Sequence, the time when the nuclear reactions start to happen.

[10]In a Lagrangian mesh, the mesh points are attached to the mass elements and they deform with them, instead of acting like a background grid.

EVtwin uses the OPAL opacities [13], and Los Alamos [22] opacities for low temperatures. For the equation of state, it uses a formula that approximates the Fermi-Dirac integrals[11] for electron density, pressure, and internal energy. It is derived in [23].

5 Modelling the Sun

In the following sections, we will present the outcome of several simulations. The goal will be to obtain information about the structure of the Sun during the Main Sequence phase and afterwards. Then be will compare the results given by MESA and EVtwin. The reproducibility instructions are in Appendix A, and the code is documented in Appendix B.

The initial structural profiles for each simulation will take the default values, differing for MESA and EVtwin. This is an uncontrolled discrepancy factor. However, customizing the r, P, L and T profiles is not an easy task. As we confirmed when preparing the simulations, fiddling with these profiles can make the solvers diverge. This arises from the Taylor expansion in the previous section (equation 15). If the initial guess is too far from the solution, the required corrections are too large, and the second-order terms become relevant. In that case the linearization yields wrong corrections.

The initial chemical composition will be custom-made, using literature values for the metal abundances in the Sun. There is an open debate around these values, with two different composition options (GS98[24] versus AGS[11]). The first one is older (1998) and predicts higher metallicity and higher C, N, and O content. The discussion occurs because the newer composition (2007) doesn't agree with helioseismological results when used in solar models. Recent (2019) solar wind data, however, point once more towards a high-metallicity Sun [25].

Due to this ongoing discrepancy we have performed two simulations with each solver, using the older and newer compositions respectively, as shown in table 1.

5.1 Initial conditions for MESA and EVtwin

MESA and EVtwin are implemented in different ways into AMUSE, and currently do not offer the same customizing opportunities. For instance, we were able to completely switch

[11]Considering the electrons as a Fermi gas, the electron density, pressure and internal energy ($f = (\rho_e, P_e, U_e)$) are given by integrals of the form $f = \int_0^\infty \frac{g(E)dE}{e^{E/T-\varphi}+1}$ where g is a different function for each quantity f, φ is an electron degeneracy parameter, E stands for energy and T for temperature. These integrals can be expanded as a polynomial in the two variables φ and T. More on this in [23].

	GS98	AGS
Z	0.01696	0.01220
H	0.74045	0.75830
He	0.24296	0.22980
C	2.90e-3	2.16e-3
N	8.49e-4	6.18e-4
O	7.88e-3	5.36e-3
Ne	1.77e-3	1.02e-3
Mg	6.73e-4	6.03e-4
Si	7.26e-4	6.66e-4
S	4.99e-4	3.24e-4
Fe	1.29e-3	1.15e-3

Table 1: Mass fractions in the two considered compositions

	MESA	EVtwin
$R(R_\odot)$	0.8779	0.8774
$R_c(R_\odot)$	0.2039	0.2066
$P_c(10^{16}Pa)$	1.4461	1.3466
$L(L_\odot)$	0.6697	0.5681
$T_c(10^7K)$	1.3200	1.2748
$T_{eff}(K)$	5.5776	5.3541

Table 2: Selected values from the initial structure.

off stellar wind in MESA but not in EVtwin. This might be due to the fact that EVtwin was originally made to calculate binary stellar evolution, where mass transfer is key. The remaining wind in our simulations follows the Watcher's [26] fitting formula, shown below.

$$\log \dot{M}_{AGB} = -4.52 + 2.74 \log(10^{-4}\frac{L}{L_\odot}) - 6.81 \log(\frac{T_{eff}}{2600K}) - 1.95 \log(\frac{M}{M_\odot}) \quad .$$

Some values from the initial structure are indicated in table 2: The total radius R, the core radius R_c, the central pressure P_c the luminosity L, and the central T_c and effective T_{eff} temperatures. We have defined the core radius as the first value containing 30% of the solar mass.

5.2 Simulations with GS98[24] composition

With this composition, the simulations with MESA and EVtwin have progressed for $12.037 Gyr$ and $12.030 Gyr$ respectively before becoming extremely slow. In both time intervals, the star remains in the Main Sequence according to AMUSE criteria.

We compared the results for the present age of the Sun with a reference, obtaining figure 5. The chosen reference is a standard solar model by Vinyoles et al. [27]. These calculations used both composition choices GS98[24] and AGS[11]. Thus, we have an appropriate benchmark for each case. The comparison for composition GS98[24] is summarized in figure 5 and table 3.

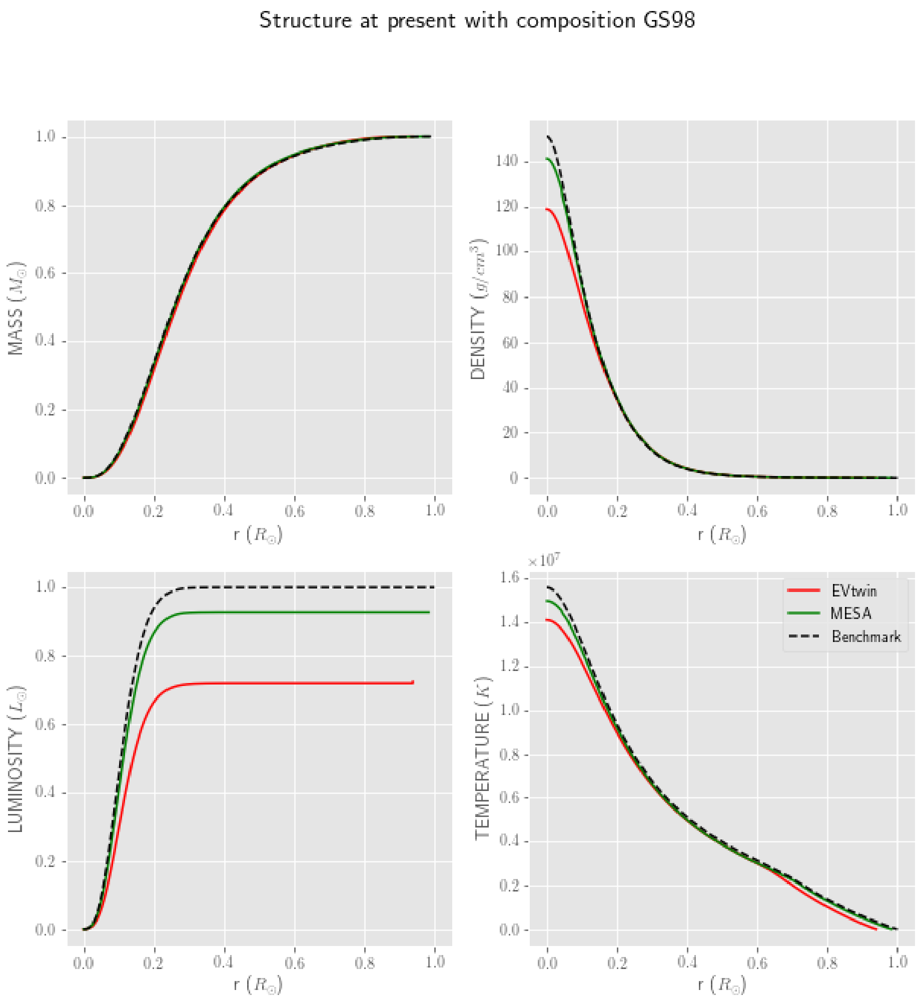

Figure 5: Comparison of the structural profiles at $4.5395 Gyr$

Let us accept the resemblance to this benchmark as a measure of the quality of the simulations. In that basis, we can state by looking at Table 3 that MESA did a better job than EVtwin: all of the relative errors in column 2 are smaller than the corresponding errors in column 4.

Table 3 shows some values of the profiles at $4.5395 Gyr$, which is the age of the Sun according to the benchmark. The second column for each simulation lists the relative error ($\Delta f = (f - f_{Bench})/f_{Bench}$).

	MESA		EVtwin		Benchmark
$R(R_\odot)$	0.9837	-0.0163	0.9393	-0.0607	1.0
$R_c(R_\odot)$	0.1914	0.0155	0.1991	0.0563	0.1885
$P_c(10^{16}Pa)$	2.1910	-0.0532	1.8770	-0.1888	2.3140
$L(L_\odot)$	0.9268	-0.0722	0.7237	-0.2755	1.0000
$T_c(10^7K)$	1.4983	-0.0396	1.4113	-0.0953	1.5600
$T_{eff}(K)$	5714	-0.0107	5497	-0.0484	5776

Table 3: Selected values from the structure at present for composition GS98[24].

As we can see in table 3, the worst determined magnitude is the luminosity, with 7.22% and 27.55% errors for MESA and EVtwin. It is worth noting that the benchmark authors use $L_\odot = 3.8418 \cdot 10^{33} erg/s$, whereas the AMUSE unit module uses $L_\odot = 3.839 \cdot 10^{33} erg/s$. This correction has been taken into account.

Let us seize the opportunity to discuss some general aspects of the Main Sequence evolution of sun-like stars. This is the longest and most stable phase for most stars. Its central feature is the conversion of Hydrogen into Helium in the core. Structural changes happen on a timescale of $1Gyr$, and we can estimate the duration of this period as $t_{nuc} \sim \frac{mc^2}{L} \sim 10Gyr$, where $m \sim 0.1M_\odot$ is the fusing mass. In the case of the Sun, fusion takes place fundamentally through the proton-proton (pp) chain. For more massive stars, the CNO cycle dominates over the pp chain.

Broadly speaking, the main-sequence evolution of a $1M_\odot$ star is distinguished by increasing central temperature and density. The luminosity and radius also increase slightly.

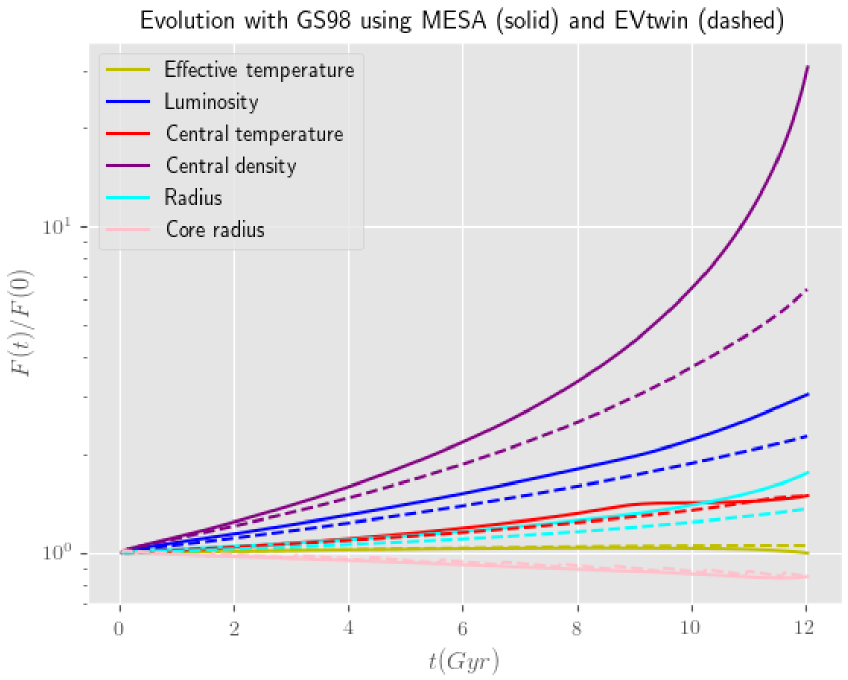

Figure 6: Normalized evolution of selected values.

All of this happens because, as Hydrogen fuses into Helium, the mean molecular weight in the core grows. Recalling the ideal gas equation of state ($P_g = \frac{K_B}{\mu m_u}\rho T$), with increasing μ, P_g will not be able to support the weight of the stellar envelope. Hence, the nucleus collapses and the central density rises. As a consequence of the Virial Theorem, half of the gravitational potential energy is delivered in the form of radiation and the other half is used to heat the core.

Furthermore, the pp chain reaction rate r_{pp} can be approximated by a power series around $T = 1.5 \times 10^7 K$ giving $r_{pp} \propto \rho X^2 \left(\frac{T}{10^6 K}\right)^4$ [8], where X is the Hydrogen mass fraction. The released power by unit mass is $\epsilon \propto r_{pp}$. Consequently, the growth in central temperature and density offset the decrease in the Hydrogen mass fraction X and the luminosity, radius and effective temperature increase slightly.

Overall, figure 6 is consistent with these trends. As we could already expect from figure 5, the luminosity and central density given by EVtwin do not increase as much as those of MESA, indicating the use of different nuclear reaction networks. This is confirmed by the snapshots of simplified mass fractions in figure 7. Despite being older, the Hydrogen mass fraction is higher in EVtwin than in MESA. Together with the failure to reach solar luminosity at the present, this shows that nuclear reactions are being underestimated in EVtwin.

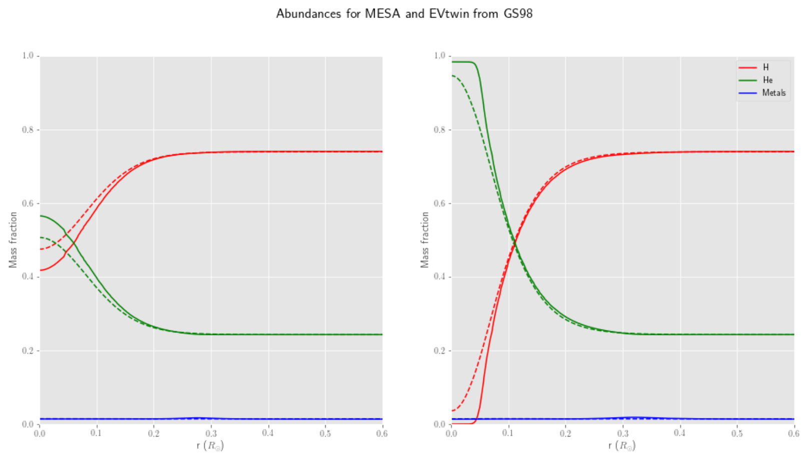

Figure 7: Mass fractions at different ages. Left: 4.54Gyr for EVtwin and 4.517Gyr for MESA. Right:11.349Gyr for EVtwin and 11.311Gyr for MESA.

	MESA		EVtwin		Benchmark
$R(R_\odot)$	1.0036	0.0036	0.9175	-0.0825	1.0
$R_c(R_\odot)$	0.1887	0.0011	0.1998	0.0601	0.1885
$P_c(10^{16}Pa)$	2.3401	0.0179	1.7115	-0.2555	2.2990
$L(L_\odot)$	1.0353	0.0364	0.6287	-0.3706	0.9989
$T_c(10^7K)$	1.5165	-0.0178	1.3543	-0.1228	1.5440
$T_{eff}(K)$	5815	0.0069	5369	-0.0704	5776

Table 4: Selected values from the structure at present for composition AGS[11].

5.3 Simulations with AGS[11] composition

In this case, with lower Z and He (and higher H), the simulations have advanced until the ages of 12.069Gyr for MESA and 12.045Gyr for EVtwin. The star in MESA has reached a higher maturity than in the previous simulation, progressing to the next stage of evolution at 11.716Gyr: The beginning of First Giant Branch. The star in EVtwin is underdeveloped and has remained in the Main Sequence.

Comparing the results for age 4.5395Gyr we obtain figure 8 and table 4. Overall, table 4 shows better results than table 3 for composition GS98[24]. Once more, MESA made better predictions than EVtwin.

Figure 9 shows the evolution of both simulations. The vertical dashed line marks the

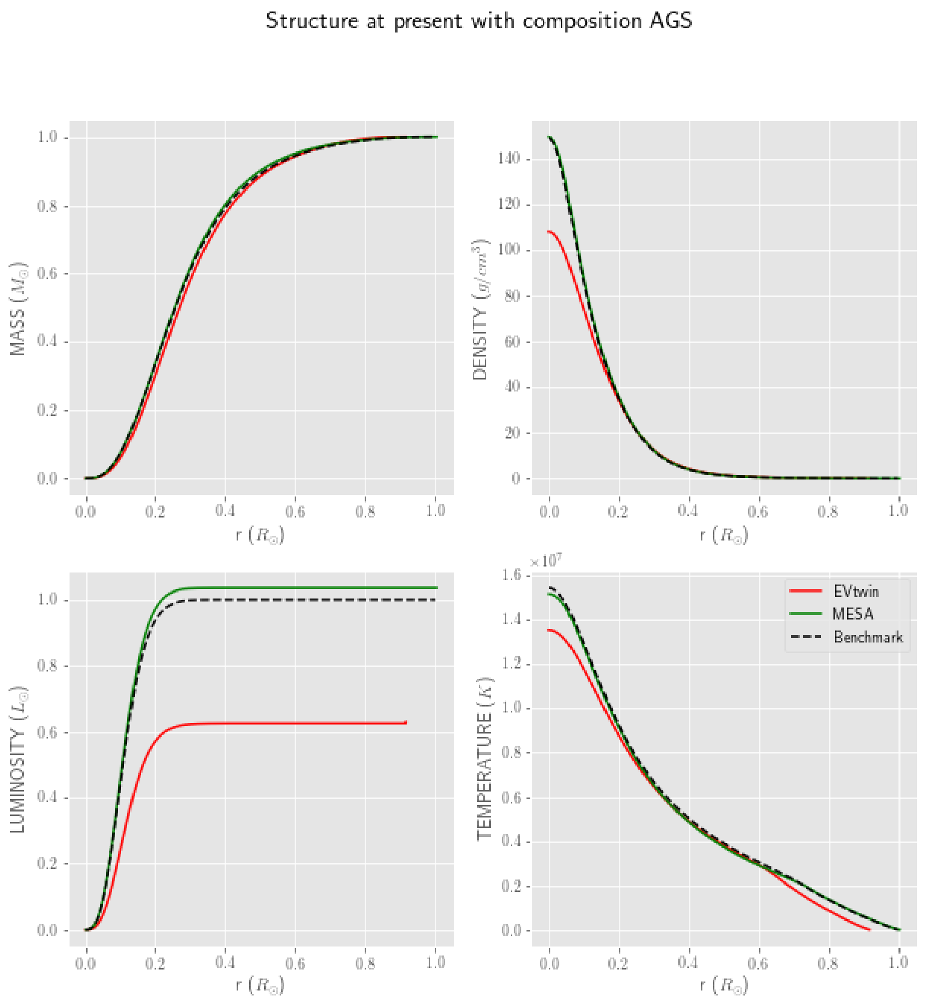

Figure 8: Comparison of the structural profiles at 4.5395Gyr

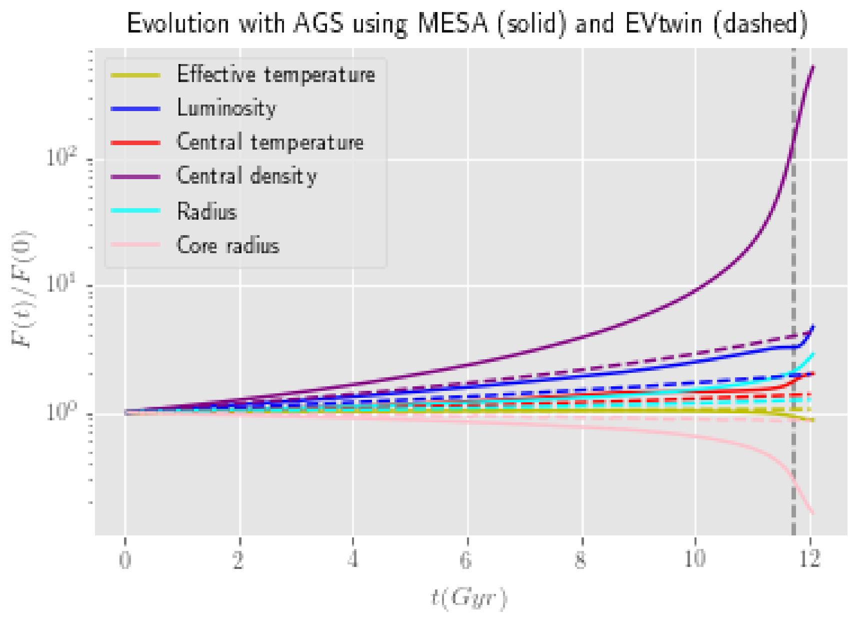

Figure 9: Evolution of selected values for initial composition AGS[11]

beginning of the First Giant Branch for MESA, with a clear change of slope for some of the magnitudes.

Let us describe the principal features of the First Giant Branch. This period begins after hydrogen exhaustion in the core. It is extremely unstable compared with the Main Sequence. When the $1M_\odot$ star enters the FGB, a region surrounding the core is hot enough for shell hydrogen fusion to begin. The H depleted nucleus now has zero luminosity and becomes isothermal. This is because the energy transport equation is $\frac{\partial T}{\partial m} \propto \nabla_{rad} \propto L$, recalling Section 3.2.4.

An isothermal core requires a strong enough pressure gradient to be able to support the outer layers, especially if its mass exceeds the Schönberg-Chandrasekhar (S-C) limit. This threshold marks the fraction of the star's mass that an isothermal nucleus can contain while maintaining hydrostatic equilibrium. It depends on the mean molecular weights of the nucleus and the envelope.

If the core exceeds the S-C limit, it will begin to contract unless a new contribution to the pressure prevents it. This can happen if the central density is so high that the electrons become degenerate. Then, they are forced to stack up in the lowest available energy levels, acquiring a non-thermal motion due to the Pauli exclusion principle. This leads to an additional pressure term, which is independent of temperature if the degeneracy is complete: $P_{deg} \propto \rho^{5/3}$ for non-relativistic electrons. In the case of the Sun, electron degeneracy becomes important during the FGB.

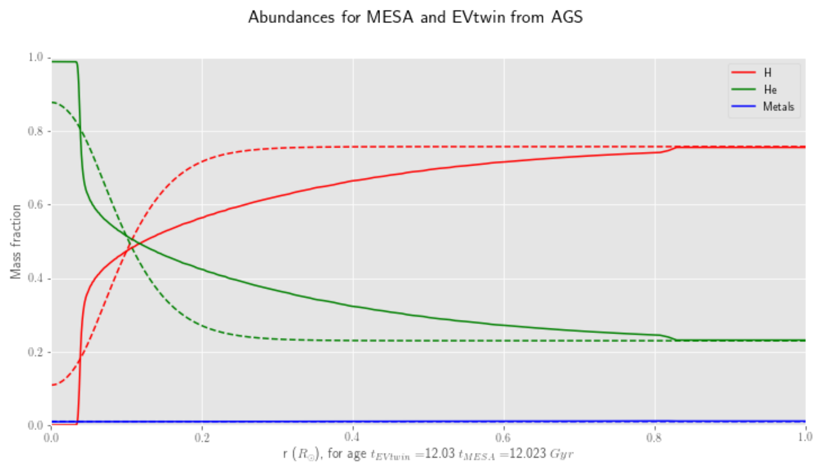

Figure 10: Mass fractions for $t_{MESA} = 12.23 Gyr$ and $t_{EVtwin} = 12.03 Gyr$

Meanwhile, H shell fusion generates more energy than in the Main Sequence because the nuclear burning region is larger. This results in an increased luminosity and a remarkable expansion of the envelope, which causes the effective temperature to drop.

Taking the core radius as the coordinate containing $0.3M_\odot$, figures showed an abrupt increase for MESA in the FGB. However, this effect was due to our definition of the core. Running the simulation again with R_c such that $M(R_c) = 0.1M_\odot$ gives figure 9, with the anticipated trend.

The rest of the curves in figure 9 go as expected for the FGB evolution. The central density increases due to contraction of the core. The slope in luminosity grows, indicating greater energy generation than in the Main-Sequence. The radius grows because the outer layers are expanding, and the effective temperature drops for the same reason. As the dashed lines for EVtwin show, the star in this simulation is extremely underdeveloped and still shows Main-Sequence behaviour.

Figure 10 shows Hydrogen shell burning in the region between $0.05 - 0.8R_\odot$, and a Helium core for $r < 0.05R_\odot$. As we can see in the dashed lines, the star in EVtwin has not even finished burning Hydrogen in the nucleus, despite being slightly older than MESA's.

	Alternative
Z	0.019
H	0.706
He	0.275
C	3.03e-3
N	1.10e-3
O	9.59e-3
Ne	1.62e-3
Mg	5.15e-4
Si	6.53e-4
Fe	1.17e-3

Table 5: Initial mass fractions for the alternative composition

5.4 An alternative composition

To end with, we have performed one last simulation with MESA. The initial composition is summarized in table 5, and the initial structure is the same as in the two previous MESA simulations. In this case, the star evolved for $11.55 Gyr$, throughout the Main Sequence and all of the FGB.

There is no benchmark available for this composition, but we have decided to include it in this work because it displays some interesting features.

The star entered the FGB at $10.58 Gyr$, as indicated by the first vertical line in figure 11. The second vertical line at $11.44 Gyr$ marks the Helium Flash, an event we are about to describe. During the Main Sequence, the curves showed the same tendencies we have already seen in the rest of the simulations. Therefore, we have skipped them for clarity.

As we have already mentioned, progressing along the FGB we have a degenerate core with rising temperature. This causes a thermal runaway because the degeneracy pressure is independent of T and the core is not expanding. Hence, there is nothing to restrain the rising temperature and the nucleus heats uncontrollably. This takes place until the core is hot enough to lift the degeneracy, allowing itself to expand.

Helium fusion begins when $T_c > 10^8 K$ via the triple-alpha process,

$$^4He +^4 He \rightarrow^8 Be$$

$$^8Be +^4 He \rightarrow^{12} C + \gamma + E$$

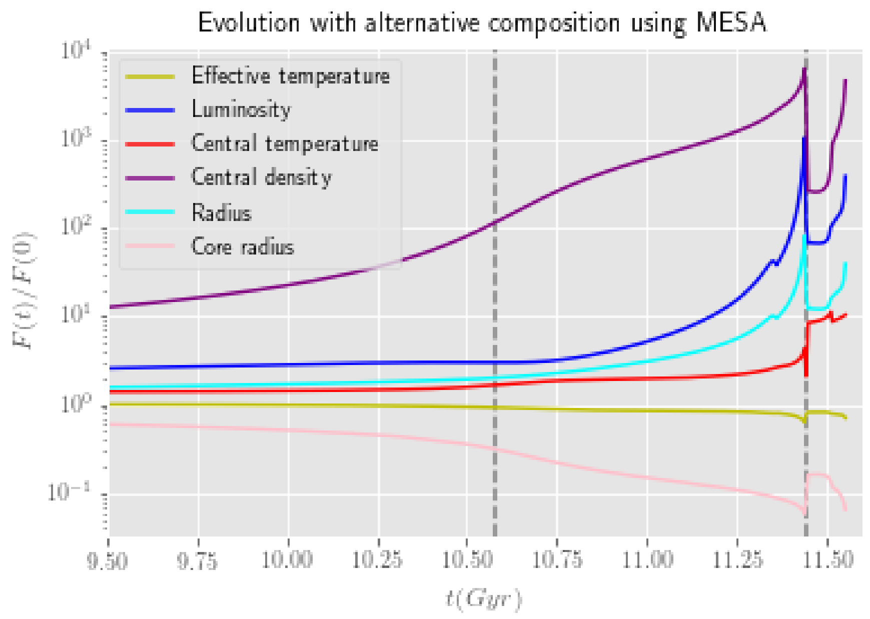

Figure 11: Evolution of selected values from alternative composition.

The energy generation rate can be approximated as $\epsilon_{3\alpha} \propto \left(\frac{T}{10^8 K}\right)^{41}$. This energy burst produces an extraordinarily high local luminosity deep inside the star, where $L \sim 10^{11} L_\odot$ (equivalent to the whole Milky Way!) for a very brief period. However, this is not appreciable outside the star because the great majority of the energy is used to abruptly expand the core: With such a high central temperature, the ideal gas pressure ($P_g \propto \rho T$) begins to dominate against the degeneracy pressure, causing the core to explode. Then the central density and temperature drop, and Helium burning continues under control. The star enters the next stage of evolution: the Horizontal Branch.

These extreme changes can be seen in figure 11, where the pink line pictures the $0.1 M_\odot$ core radius. The central temperature first reached $10^8 K$ at the age of $11.44 Gyr$, and we immediately saw the beginning of Helium burning. Figure 11 shows the drastic decrease in central density and increase in core radius corresponding to the explosion of the nucleus. The luminosity and the radius present a narrow peak, decreasing in the Helium Flash. The central temperature drops briefly (as expected), but grows again after the Flash.

The composition snapshots in figure 12 show Hydrogen shell burning ($10.885 Gyr$), contraction of this shell ($11.154 Gyr$), and the onset of Helium fusion with an expansion of the core ($11.458 Gyr$). This agrees with the expectations we have discussed.

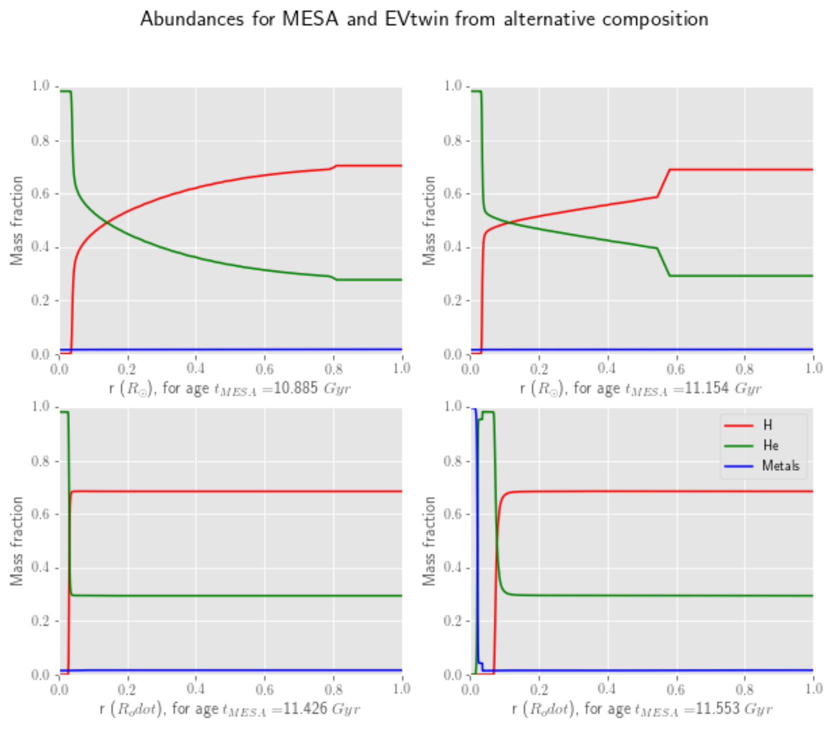

Figure 12: Snapshots of the mass fractions in the alternative simulation.

Finally, figure 13 shows the Hertzsprung-Russell diagram for this simulation. It is qualitatively very similar to the track of the Sun in the HR plot. The lowest luminosity part corresponds to the Main Sequence, and then the star progresses through the FGB with increasing L. The Helium flash is located at the tip of the FGB, where the luminosity reaches $1000L_\odot$.

Part III
Conclusions

As stated in the introduction, in this work we have presented the equations of stellar structure and evolution (Section 3) and the most common method to solve them (The Henyey method, in Section 4). In brief, we have seen that the Henyey method is based on a linearization of the equations of stellar structure and evolution. Starting from an initial guess for all the variables, we solve the linearized system iteratively to find corrections that ultimately lead to a converged model. Then we update the chemical composition,

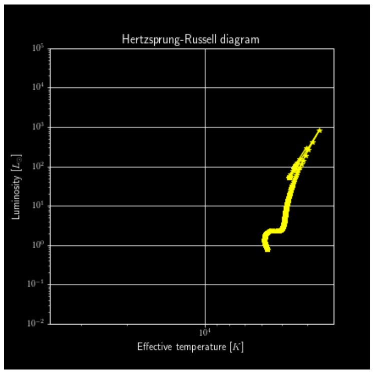

Figure 13: HR diagram for the alternative simulation

which drives the stellar evolution.

In Section 5 we have presented the outcome of several simulations, performed with two different implementations of the Henyey method: MESA and EVtwin (previously described in sections 4.2.1 and 4.2.2). These implementations are included in the AMUSE software environment. We have introduced this software and provided a tutorial in Section 2. We supply the code of the tutorial and the simulations as complementary material, with the corresponding documentation in Section 2 and Appendix B.

Due to the ongoing debate about solar metal content, we have implemented simulations with MESA and EVtwin for two different compositions. For each mixture, we have compared the results of the two implementations with a benchmark. In both cases, MESA achieved a greater resemblance to the reference. Given the complexity of the stellar evolution codes, we cannot give a single reason for this discrepancy. However, the use of different nuclear reaction networks may be one of the dominant factors. This claim is supported by the underdevelopment observed in the mass fractions from EVtwin compared to MESA (figures 7 and 10). We have also tracked the evolution of a set of selected values, and we have seen in figures 6 and 9 that they verify the trends predicted in the literature.

Finally, we have performed one last simulation with an alternative composition. This time the calculations advanced until the end of the First Giant Branch, including the Helium Flash. We have observed the effects of this event in figure 11.

Despite the simplicity of the theory presented in Section 4 (Numerical Methods), implementing these ideas into working code is far from trivial. The initially naive picture starts to break when reading the descriptions of MESA and EVtwin in sections 4.2.1 and 4.2.2. However, this work has covered nothing more than the tip of the iceberg. For instance, constructing appropriate grids can be a key source of difficulties. Another challenge is finding suitable initial models. An important matter we have left untreated is the propagation of numerical errors.

As to the simulations in Section 5, there are a handful of ideas that could potentially improve the results. For example, it is a common practice to calibrate the solar models by varying the mixing length ratio α until the model reflects the observed properties of the Sun. In all of our simulations, we have simply used the default $\alpha = 2$.

Another possible improvement would be to use the same initial structure for the simulations with MESA and EVtwin. Still, due to the very different number of grid points in the two implementations, this is not so easy. A potential workaround is to choose the default EVtwin structure and generate the extra grid points by interpolation.

Following the literature, we have affirmed that the composition GS98[24] reproduces helioseismological results better than AGS[11], but we have not checked this in our simulations. To do this, we would have to compute the sound velocity profiles for the stellar model of age $4.5395 Gyr$ and compare them with a reliable source of experimental data.

Finally, we have observed that in both implementations the ability to evolve stellar models for a long time is dependent on composition. All of the simulations have progressed until the calculations became too time-consuming. This might be due to our limited resources, and it could change with parallel computing. Nonetheless, we could investigate this composition-dependence, preparing a set of additional simulations with systematically varying mass fractions.

References

[1] N. de Vries, S. Portegies Zwart, and J. Figueira, "The evolution of triples with a roche lobe filling outer star," *Monthly Notices of the Royal Astronomical Society*, vol. 438, no. 3, 1909–1921, 2014, ISSN: 1365-2966. DOI: 10.1093/mnras/stt1688. [Online]. Available: http://dx.doi.org/10.1093/mnras/stt1688.

[2] M. P. Forum, "Mpi: A message-passing interface standard," USA, Tech. Rep., 1994.

[3] J. Maassen and H. E. Bal, "Smartsockets: Solving the connectivity problems in grid computing," in *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, ser. HPDC '07, Monterey, California, USA: Association for Computing Machinery, 2007, 1–10, ISBN: 9781595936738. DOI: 10.1145/1272366.1272368. [Online]. Available: https://doi.org/10.1145/1272366.1272368.

[4] S. Portegies Zwart and S. McMillan, *Astrophysical Recipes*, ser. 2514-3433. IOP Publishing, 2018, ISBN: 978-0-7503-1320-9. DOI: 10.1088/978-0-7503-1320-9. [Online]. Available: http://dx.doi.org/10.1088/978-0-7503-1320-9.

[5] S. McMillan, A. van Elteran, and A. Whitehead, *Astronomical Society of the Pacific Conference Series*, vol. 453, 129, 2012.

[6] T. R. Carson, D. F. Mayers, and D. W. N. Stibbs, "The calculation of stellar radiative opacity,", vol. 140, p. 483, Jan. 1968. DOI: 10.1093/mnras/140.4.483.

[7] R. Kippenhahn, A. Weigert, and A. Weiss, *Stellar Structure and Evolution.* 2012. DOI: 10.1007/978-3-642-30304-3.

[8] B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics*, 2nd ed. Cambridge University Press, 2017. DOI: 10.1017/9781108380980.

[9] P. Bodenheimer, *Numerical Methods in Astrophysics: An Introduction.*

[10] L. G. Henyey, L. Wilets, K. H. Böhm, R. Lelevier, and R. D. Levee, "A Method for Automatic Computation of Stellar Evolution.,", vol. 129, p. 628, May 1959. DOI: 10.1086/146661.

[11] M. Asplund, N. Grevesse, and A. J. Sauval, "The Solar Chemical Composition,", vol. 130, no. 1-4, pp. 105–114, Jun. 2007. DOI: 10.1007/s11214-007-9173-7.

[12] B. Paxton, L. Bildsten, A. Dotter, F. Herwig, P. Lesaffre, and F. Timmes, "Modules for experiments in stellar astrophysics (mesa)," *The Astrophysical Journal Supplement Series*, vol. 192, no. 1, p. 3, 2010, ISSN: 1538-4365. DOI: 10.1088/0067-0049/192/1/3. [Online]. Available: http://dx.doi.org/10.1088/0067-0049/192/1/3.

[13] C. A. Iglesias and F. J. Rogers, "Updated Opal Opacities,", vol. 464, p. 943, Jun. 1996. DOI: 10.1086/177381.

[14] D. Saumon, G. Chabrier, and H. M. van Horn, "An Equation of State for Low-Mass Stars and Giant Planets,", vol. 99, p. 713, Aug. 1995. DOI: 10.1086/192204.

[15] C. Angulo, M. Arnould, M. Rayet, P. Descouvemont, D. Baye, C. Leclercq-Willain, A. Coc, S. Barhoumi, P. Aguer, C. Rolfs, R. Kunz, J. Hammer, A. Mayer, T. Paradellis, S. Kossionides, C. Chronidou, K. Spyrou, S. Degl'Innocenti, G. Fiorentini, B. Ricci, S. Zavatarelli, C. Providencia, H. Wolters, J. Soares, C. Grama, J. Rahighi, A. Shotter, and M. L. Rachti], "A compilation of charged-particle induced thermonuclear reaction rates," *Nuclear Physics A*, vol. 656, no. 1, pp. 3 –183, 1999, ISSN: 0375-9474. DOI: https://doi.org/10.1016/S0375-9474(99)00030-5. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0375947499000305.

[16] G. R. Caughlan and W. A. Fowler, "Thermonuclear reaction rates v," *Atomic Data and Nuclear Data Tables*, vol. 40, no. 2, pp. 283 –334, 1988, ISSN: 0092-640X. DOI: https://doi.org/10.1016/0092-640X(88)90009-5. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0092640X88900095.

[17] S. Cassisi, A. Y. Potekhin, A. Pietrinferni, M. Catelan, and M. Salaris, "Updated electron-conduction opacities: The impact on low-mass stellar models," *The Astrophysical Journal*, vol. 661, no. 2, 1094–1104, 2007, ISSN: 1538-4357. DOI: 10.1086/516819. [Online]. Available: http://dx.doi.org/10.1086/516819.

[18] J. Iben I., "Thermal pulses: p-capture, alpha -capture, s-process nucleosynthesis; and convective mixing in a star of intermediate mass.,", vol. 196, pp. 525–547, Mar. 1975. DOI: 10.1086/153433.

[19] J. W. Ferguson, D. R. Alexander, F. Allard, T. Barman, J. G. Bodnarik, P. H. Hauschildt, A. Heffner-Wong, and A. Tamanai, "Low-temperature opacities," *The Astrophysical Journal*, vol. 623, no. 1, pp. 585–596, 2005. DOI: 10.1086/428642. [Online]. Available: https://doi.org/10.1086%2F428642.

[20] Z. Han, P. Podsiadlowski, and P. P. Eggleton, "A possible criterion for envelope ejection in asymptotic giant branch or first giant branch stars," *Monthly Notices of the Royal Astronomical Society*, vol. 270, no. 1, pp. 121–130, Sep. 1994, ISSN: 0035-8711. DOI: 10.1093/mnras/270.1.121. eprint: https://academic.oup.com/mnras/article-pdf/270/1/121/3889699/mnras270-0121.pdf. [Online]. Available: https://doi.org/10.1093/mnras/270.1.121.

[21] P. P. Eggleton, "The Evolution of Low Mass Stars," *Monthly Notices of the Royal Astronomical Society*, vol. 151, no. 3, pp. 351–364, Feb. 1971, ISSN: 0035-8711. DOI: 10.1093/mnras/151.3.351. eprint: https://academic.oup.com/mnras/article-pdf/151/3/351/9360845/mnras151-0351.pdf. [Online]. Available: https://doi.org/10.1093/mnras/151.3.351.

[22] A. Weiss, J. Keady, and N. Magee, “A collection of los alamos opacity tables for all temperatures,” *Atomic Data and Nuclear Data Tables*, vol. 45, no. 2, pp. 209 –238, 1990, ISSN: 0092-640X. DOI: https://doi.org/10.1016/0092-640X(90)90008-8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0092640X90900088.

[23] P. P. Eggleton, J. Faulkner, and B. P. Flannery, “An Approximate Equation of State for Stellar Material,”, vol. 23, p. 325, Mar. 1973.

[24] N. Grevesse and A. J. Sauval, “Standard Solar Composition,”, vol. 85, pp. 161–174, May 1998. DOI: 10.1023/A:1005161325181.

[25] S. Vagnozzi, “New solar metallicity measurements,” *Atoms*, vol. 7, no. 2, E. Augé, J. Dumarchez, and J. Trân Thanh Vân, Eds., p. 41, 2019. DOI: 10.3390/atoms7020041. arXiv: 1703.10834 [astro-ph.SR].

[26] A. Wachter, K.-P. Schroder, J. Winters, T. Arndt, and E. Sedlmayr, “An improved mass-loss description for dust-driven superwinds and tip-AGB evolution models,” *Astron. Astrophys.*, vol. 384, pp. 452–459, 2002. DOI: 10.1051/0004-6361:20020022.

[27] N. Vinyoles, A. M. Serenelli, F. L. Villante, S. Basu, J. Bergström, M. C. Gonzalez-Garcia, M. Maltoni, C. Peña-Garay, and N. Song, “A new generation of standard solar models,” *The Astrophysical Journal*, vol. 835, no. 2, p. 202, 2017, ISSN: 1538-4357. DOI: 10.3847/1538-4357/835/2/202. [Online]. Available: http://dx.doi.org/10.3847/1538-4357/835/2/202.