400A - Neutrinos & computational stellar evolution

Mathieu Renzo

February 27, 2025

Materials: Onno Pols' notes Sec. 6.5 and 12.3.1, Chapter 7, Kippenhahn chapter 9, 10, 11

Neutrino physics in stars

We have almost seen all the pieces needed to make a one-dimensional hydrostatic model of a star, including the input physics in the quantities κ (opacity) and $\varepsilon_{\mathrm{nuc}}$. The aim of this lecture is to unpack the last input quantities entering in the energy conservation which connects stellar physics with neutrino physics: ε_{ν}.

Summary of equations we have derived

Mass conservation

$$\frac{dr}{dm} = \frac{1}{4\pi r^2 \rho} \quad . \tag{1}$$

Hydrostatic equilibrium

$$\frac{dP}{dm} = -\frac{Gm}{4\pi r^4} \quad , \tag{2}$$

which follows from the momentum conservation equation assuming the acceleration to be negligible **N.B.:** this last assumption could be relaxed to add an extra term in this equation analogous to ma in F=ma.

Equation of state

$$P_{\rm tot} = P_{\rm gas} + P_{\rm rad} = \frac{\rho}{\mu m_u} k_B T + P_{QM} + \frac{1}{3} a T^4 \quad . \tag{3}$$

Energy transport

$$\frac{dT}{dm} = \frac{T}{P}\frac{dP}{dm}\nabla \tag{4}$$

where $\nabla = \partial$ log(T)/∂ log(P) is the local temperature gradient, equal to the radiative gradient in stably stratified regions:

$$\nabla \equiv \nabla_{\rm rad} = \frac{3P}{14\pi acGmT^4}\kappa L \tag{5}$$

with $\kappa = (1/\kappa_{\rm rad} + 1/\kappa_{\rm cond})^{-1}$ the combination "in parallel" of the radiative and conductive opacity (assumed to be known from atomic physics), and $\nabla \equiv \nabla_{\rm ad}$ the adiabatic gradient (within $\sim 10^{-7-8}$ precision) for convective regions. We also have a criterion (Schwarzschild criterion) to determine which region is which.

Energy conservation

$$\frac{dL}{dm} = \varepsilon_{\rm nuc} - \varepsilon_\nu + \varepsilon_{\rm grav} \quad . \tag{6}$$

with $\varepsilon_{\rm grav}$ = T∂ s/∂ t the change in internal energy and $\varepsilon_\nu > 0$ ($\Rightarrow$ it is always a *loss* term during stellar evolution).

Composition evolution

$$\frac{dX_i}{dt} = A_i \frac{m_u}{\rho}\left(\sum_{k,l} r_{k,l} - \sum_{i,j}(1+\delta_{ij})r_{ij}\right) + D_{\rm mix}\frac{dX_i}{dr} \equiv \frac{dX_i}{dt}(T,\rho,X_j) \quad , \tag{7}$$

where the first two terms represent production and destruction of the i-th isotope by nuclear processes and the third term represent mixing by various instabilities treated in diffusion approximation.

Two types of neutrinos

For the density typical in stellar interior (think of $\langle\rho_\odot\rangle$ and the other average densities you estimated in the various homeworks), neutrinos can safely be considered as non-interacting. This means we can consider neutrinos as ultra-relativistic mass-less particles propagating at $v \simeq c$. The mass less approximation also implies that we neglect neutrino oscillations: even if a neutrino changes leptonic flavor, it's still going to leave the star carrying away its energy!

N.B.: recall that the leptonic number is +1 for the leptons electron e^-, muon μ^-, tau τ^- and the corresponding neutrinos ν_e, ν_μ, ν_τ and -1 for their antiparticles positron e^+, positive muon μ^+, and positive τ^+ and the corresponding antineutrinos.

Therefore, any neutrino produced in the stellar interior leaves the star in the light-crossing time $R/c \leq 1000\ R_\odot/c \simeq 1000 \times 2$ sec $= 2000$ sec which is much shorter than evolutionary timescales (and even during shorter evolutionary phases such as silicon shell burning, the energy carried by the neutrinos is on its way out and nothing is going to stop it from leaving the star).

These arguments apply to both kinds of neutrinos that we encounter in stellar evolution (and are about to introduce), and stop working only when densities get extremely high ($\rho \geq 10^{10}$ g cm^{-3}): at this point neutrinos can and do start interacting with matter. Such densities are only encountered in a collapsing core that has exhausted nuclear fuel and in neutron stars: neutrino interactions are crucial to the physics of the supernova explosions, but can safely be neglected before.

During the evolution of the star that sets the stage for the explosion, we encounter two "classes" of neutrinos.

Nuclear neutrinos

We have already encountered processes producing neutrinos occurring in stars: *weak nuclear reactions*! These are needed since the main sequence (both in the pp chain and CNO) to convert some protons into neutrons for the helium nuclei. On the scale of r_{nuc}, the neutrino crossing time (that is the light crossing time) is so short that we can neglect neutrino oscillations and assume conservation of leptonic number to write down the equations.

These neutrinos do carry away energy as described above, *but* that is just some of the energy that the exo-energetic thermonuclear reaction would have released. Effectively, they decrease the energy gain from releasing nuclear binding energy but they are not a net energy loss.

This allows to incorporate them in models by re-writing $\varepsilon_{\mathrm{nuc}} \rightarrow \varepsilon_{\mathrm{nuc}}$ - $|\varepsilon_{\nu,\ \mathrm{nuc}}|$, where the neutrino term is always negative (hence we can write it as minus an absolute value), and needs to be calculated from nuclear physics, determining the spectrum of the neutrino produced by a given reaction (a non-trivial task which stellar physics leaves to neutrinos and nuclear physicists).

Particularly important for the late evolution of massive stars are neutrinos from the so-called *URCA processes* (named after a casino in Rio de Janeiro because when these processes kick in the energy of the nuclear reaction goes the same way as the money in the casino!):

$$^{A}Z + e^{-} \rightarrow^{A} (Z-1) + \nu_e{}^{A}(Z-1) \rightarrow^{A} Z + e^{+} + \bar{\nu_e} \tag{8}$$

which produce one neutrino and one anti-neutrino without changing the composition of the star. This requires that the nucleus $^{A}(Z-1)$ is unstable to β^{-}-decay and the cross section for electron capture on ^{A}Z is non-negligible, which can happen during Si core burning and the subsequent gravitational collapse of the core once the nuclear fuel runs out.

The nuclear neutrinos are mostly sensitive to the core temperature for the activation of certain thermonuclear reaction chains (except for pycno-nuclear reaction at extremely high densities where the electron screening effectively makes the Coulomb barrier negligible).

Thermal neutrinos

After helium core burning, the density in the stellar cores become sufficiently high (because of the gravothermal collapse) that non-nuclear processes producing neutrinos start occurring. After carbon depletion, the neutrinos produced by these processes can take away more energy than is locally lost to photons by each stellar layer: *evolved massive stars are neutrino stars* $\mathrm{L}_{\nu} \gg \mathrm{L}_{\mathrm{rad}}$ (Fraley 1968).

This also effectively means that the stellar core of evolved ($\sim$ during and after carbon core burning) massive stars is *decoupled* from the stellar envelope: the *gravothermal collapse of the core occurs to compensate the neutrino losses from the core itself*! The thermal timescale of the core becomes $\tau_{\mathrm{KH},\nu} \simeq \mathrm{GM}_{\mathrm{core}}{}^{2}/(2\mathrm{R}_{\mathrm{core}}\ \mathrm{L}_{\nu})$ and the nuclear timescale becomes $\tau_{\mathrm{nuc},\nu} = \phi$ $\mathrm{f}_{\mathrm{burn}}\ \mathrm{Mc}^{2}/\mathrm{L}_{\nu}$ both of which are much shorter than the timescales in the low density, photon-cooled envelope: in the late stages of stellar evolution the envelope should be *frozen* and the core evolves driven by neutrino losses.

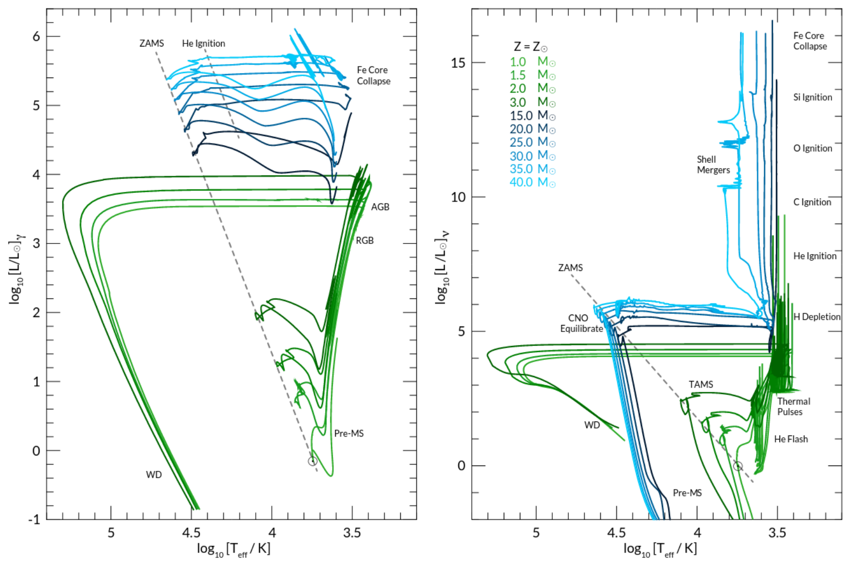

Figure 1: Left: evolutionary tracks on a photon HR diagram. Right: corresponding evolutionary tracks on a "neutrino" HR diagram. This is Fig. 2 from Farag et al. 2020. Note the y-axis scale in both panels.

N.B.: it is still the energy losses driving the gravothermal collapse because of the virial theorem that govern the evolution, but the envelope does not have time to keep up with the core.

N.B.: recently, observations of early signals of stellar explosion have questioned this picture of *frozen* envelope: there *may* be some presently unknown phenomena happening on a *dynamical* timescale of the envelope in the final years/months of a massive star evolution that affect the envelope. The fact that they *need to be dynamical* to do anything is related to the fact that the evolution of the core is sped up by the thermal neutrino losses.

The neutrinos that do *not* come from nuclear reactions are a real energy *loss* term for the star that enter in the local energy conservation equation ε_ν (which is also always negative!): dL/dm $= \varepsilon_{\mathrm{nuc}}$ - $|\varepsilon_\nu|$ $+\varepsilon_{\mathrm{grav}}$.

The *thermal* processes producing these neutrinos typically will produce a neutrino-antineutrino *pair* to conserve the leptonic number. Typically only electron neutrinos will be relevant: leptons other than e$^\pm$ are unstable and are not commonly found in stars. They fall into several categories (here x=$\{$e,μ, or $\tau\}$ but in stellar conditions it's typically e):

- plasmon processes: $\gamma + \tilde{\gamma} \rightarrow \nu_x + \bar{\nu}_x$,

- bremsstrahlung: $e^- +^A Z \rightarrow e^- +^A Z + \nu_x + \bar{\nu}_x$,
- pair-production: $\gamma + \gamma \rightarrow \nu_x + \bar{\nu}_x$,
- pair annihilation: $e^+ + e^- \rightarrow \nu_x + \bar{\nu}_x$,
- photo-processes: $\gamma + e^- \rightarrow e^- + \nu_x + \bar{\nu}_x$,

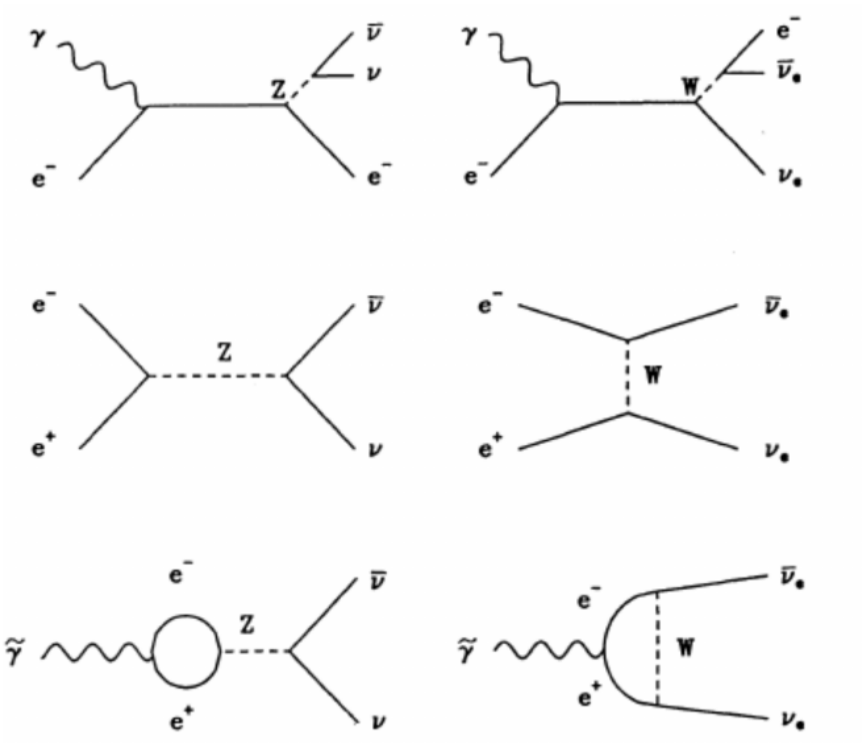

Figure 2: Feynman diagrams for the dominant neutrino cooling processes. The top row shows the photo-emission processes, the middle row shows the e± annihilation processes, the bottom row shows the plasmon processes. The neutral (charged) current ractions are in the left (right) column. This is Figure 1 from Aufderheide 1993, and Z and W represent the boson that mediate weak interactions: the left column shows interactions mediated by the neutral boson Z, while the right column shows interactions mediated by the charged boson W$^{\pm}$.

N.B.: "plasmons" are collective excitations of the stellar plasma that propagate (the analogy is with "solitons" in fluid dynamics), and can be quantized and coupled via quantum electrodynamics to e$^{\pm}$.

N.B.: The Feynman diagrams of some of these processes above are illustrating the processes (and the various pieces that contribute in the quantum field theory calculation of the cross section), the particles that are not "free legs" are mediators and not real particles, do not over-interpret these diagrams as physical pictures!

Neutrino cooling processes are mostly sensitive to the core density ρ. The typical energy carried away per neutrino-antineutrino pair is of the order of the thermal energy of the electrons (i.e., their Fermi energy if the region from where the neutrinos are emitted is partially degenerate).

Lke κ and $\varepsilon_{\mathrm{nuc}}$, the energy losses to thermal neutrinos are usually tabulated in stellar physics codes (see especially the widely used Itoh et al. 1996), and the figure below shows the $|\varepsilon_\nu| \equiv |\varepsilon_\nu|(\mathrm{T},\rho)$ resulting from these tables:

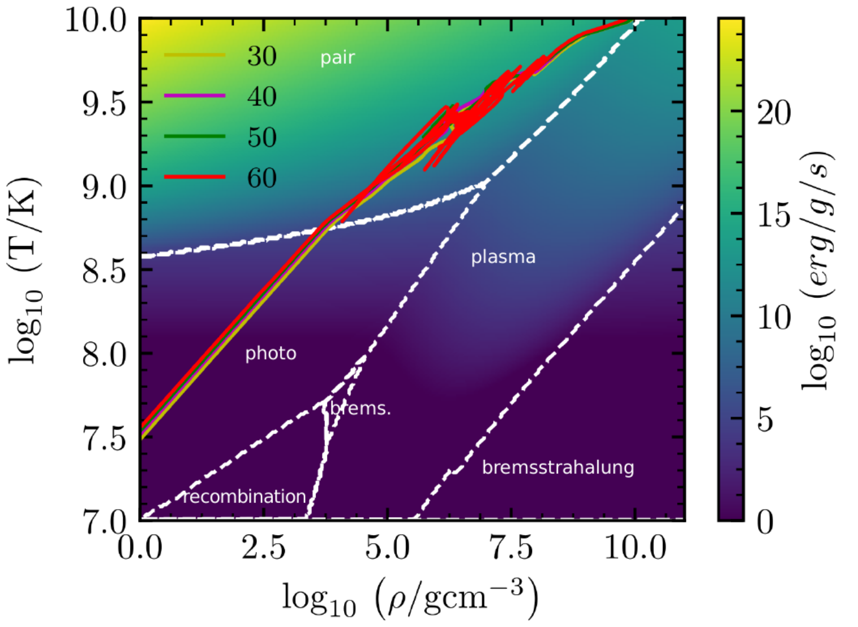

Figure 3: Neutrino energy losses on the T(ρ) plane. White lines mark the separation between regions where different neutrino emission processes dominate, the colored lines mark the T(ρ) tracks of helium cores of the labeled masses (in $\mathrm{M}_\odot$ units) computed with MESA. Credits: R. Farmer.

Principles of computational stellar evolution

"*Traditional scientific knowledge has generally taken the form of either theory or experimental data. However, where theory and experiment stumble, simulations may offer a third way.*" - Simulation, Johannes Lenhard et al.

We now have finally derived/discussed all the pieces of physics necessary to compute a stellar *structure* model, treat its nuclear energy generation and thus driving its *evolution*.

The description we have obtained for a spherically symmetric star is made of **four non-linear, coupled, ordinary differential equations** (ODE) **plus the equation of state that acts as closure condition** for the system. Solving these equations we can:

1. study the interior structure of modelled stars and try to learn about the parts of the star that are not accessible to direct observations (hidden inside the photosphere)

2. study the time evolution of modelled stars, which we cannot observe for real stars since most evolve way too slow for us to follow within our lifetimes.

However, this system of coupled, non-linear ODEs is not easily solved by hand. Since the early days of the availability of computers in the 1960s, people have been designing and leveraging *computational techniques* to *numerically* solve this system of equations (see e.g., Henyey 1959, Iben & Erhman 1962, Sujimoto 1970 Eggleton 1971, Weaver et al. 1978). For the rest of this lecture, we are going to discuss some general principles behind these computational techniques.

The most important thing: *Computer simulations are not empirical evidence*

Computational techniques take a system of equations describing a physical model (for example the equations we have derived), which already rely on a whole variety of physical approximations (e.g., spherical symmetry, LTE, free-streaming neutrinos, etc.), and apply a whole new set of *numerical* approximations to obtain a numerical solution.

Presumably, nature does not do *any* of this: without opening the philosophical question of whether nature writes down equations to solve, it certainly does not need to make physical approximations (while we need to do it to reason on a problem and keep it manageable), and even less make *numerical* approximations needed to solve equations. At ontological level, numerical simulations are *not* equivalent to empirical evidence! Never trust numerical results as if they were ground truth: a computer will only do what it is told, and we tell it our physically and numerically approximated best guess for what we are trying to simulate which is *not* what nature empirically provides. I emphasize this as a person who spends most of his time making numerical simulations! Note also how this is more general than *just* stellar physics: this applies to *any* computational physics field.

Because of these, it is always crucially important to *do resolution tests*: when performing a simulation of a physical phenomenon, you should always test that the scientific results you obtain do not depend on how you discretize your equations in time and space and on how you numerically solve them. This is often a painful task, but very important to not fool ourselves!

"*All models are wrong, but some are useful!*" - G. Box

In stellar physics, there is another problem: the *high non-linearity* of the coupled system of ODEs means that there can be chaotic behavior! A small perturbation (maybe because of a numerical error or a small inconsistency in the tabulated input physics!) can cascade into dramatic consequences, similar to the famous "butterfly effect" (which was also theorized from numerical computations, but of atmospheric physics, by E. Lorenz).

Why did we limit ourselves to spherical symmetry?

Throughout the course so far we have explicitly assumed spherical symmetry, although since the beginning we have discussed some phenomena that can break the global spherical symmetry (e.g., rotation, magnetic fields, or the presence of gravity and/or irradiation from companion stars), and we have seen phenomena (e.g., convection) that break the spherical symmetry locally. Without the assumption of spherical symmetry, the equations describing our system would become partial differential equations (PDE), changing the mathematical and thus computational approach.

In some cases, it is possible to treat some of the non-spherical effects by casting them in a form that still allows a 1D formalism with ODEs (e.g., "shellular rotation" assuming that all quantities are constant along isobars and uses P as the independent coordinate, e.g., Maeder & Meynet 2000, or using volume-weighted equivalent potentials accounting for the gravity of a companion star, e.g., Paxton et al. 2015). But ultimately the physics is non-spherical, so why do we insist with this limited approximation?

The main reason is *necessity* (see for example Jermyn et al. 2022 section 5.4) **the contrast of scale in both space and time** required to follow the *evolution* of a star from birth to death (or to the current age of the Universe, whichever comes first!) **is just too large** to be manageable with present day *and* with foreseeable computational capabilities.

Spatial scale limitations

Consider the convective envelope in the Sun. It has a Reynolds number Re$=$ $\mathrm{v_{conv}}\ \ell/\nu \sim 10^{12}$, which, using Kolmogorov's model of turbulence implies that the turbulent cascade spans a range of scales from L to ℓ with a contrast L$/\ell$ $\sim$ Re$^{3/4} \sim 10^9$. To resolve this contrast in a 3D simulation we would need this number of cells *in each direction*, meaning $\sim$$10^{27}$ points.

Temporal scale limitations

Consider again the Sun. The free fall timescale is generously $\tau_{\mathrm{ff}}$$\sim$1h, and its thermal timescale is $\tau_{\mathrm{KH}}$$\sim$$1.5\times10^7$ years, and current age is $\sim 5\times10^9$ years $\simeq$ τ_{nuc}. In 3D we would need to resolve the dynamics of the gas, with timesteps Δ t $\leq \tau_{\mathrm{ff}}$. Even considering an equal sign Δ t$=\tau_{\mathrm{ff}}$ (certainly insufficient to *resolve* the dynamics of the stellar plasma), it would take $\sim \tau_{\mathrm{Kh}}\ /\ \tau_{\mathrm{ff}} = 10^{10}$ timesteps to calculate a thermal timescale and $\sim \tau_{\mathrm{nuc}}\ /\ \tau_{\mathrm{ff}} = 10^{12}$ timesteps for a nuclear timescale.

Present day and future prospects

The largest present day multi-dimensional stellar hydrodynamics simulations reach maybe up to 10^{12} resolution points, and about 10^8 timesteps and using lower-than-realistic Reynolds number and/or boosting the luminosity to drive dynamical effects faster than in reality (see Anders et al. 2022 or Thompson et al. 2024).

Assuming that Moore's law (the number of transistors in a CPU roughly doubles every year) - an assumptions that may clash with the engineering reality, it will take several decades to have a numerically resolved 3D simulation of convection in a star for a convective turnover timescale, and longer for thermal and nuclear timescale: 1D will remain necessary, and we need to keep representing the complex multi-scale, multi-physics, and multi-dimensional problem of stellar evolution considering only variations along the radial direction!

Nevertheless, 3D stellar hydrodynamics of restricted problems ("box in a star" approach - e.g., Rizzuti et al. 2024, short timescales such as stellar explosions - e.g., Fields & Couch 2021) are possible, and stellar *structure* simulations in 2D are also at the forefront of possibilities (see e.g., Mombarag et al. 2023).

Boundary conditions

So, for the foreseeable future we will need 1D stellar structure and evolution calculation using the set of equations we derived. To solve them, we need boundary conditions. These are crucial in determining the solution of the structure at each timestep!

For the stellar problem, we need to specify boundary conditions at two locations.

Center

These are fairly intuitive:

- r(m=0) = 0. This is necessary to keep the local density $\rho \sim \mathrm{m/r^3}$ finite.
- L(m=0) = 0. This is necessary to keep the energy generation per unit volume finite (as the volume goes to zero)

However, we have 4 ODEs and these are only two boundary conditions, so the system is still undetermined. The central pressure and density are not a priori known (we can estimate them, but that is not precise enough!).

Surface

We need to turn to the surface to get observationally informed boundary conditions. Remember that from the point of view of a detailed stellar evolution code, which assumes LTE to solve for the internal structure, the "surface" (meaning: the outer boundary) is usually defined at the photosphere. This is by definition the "idealized" surface where $\mathrm{T=T_{eff}}$ which corresponds to the location outside of which LTE is not a good assumption anymore, the radiation field is *not* isotropic, and the problem becomes the calculation of a stellar *atmosphere*.

Accepting to externalize the problem of stellar atmosphere (which we will treat in more detail in a future lecture), we already have written one of the missing outer boundary conditions at mass coordinate equal to the total mass of the star (m=M):

- $\mathrm{T(m{=}M)} = \mathrm{T_{eff}}$, where $\mathrm{T_{eff}}$ is *defined* as the temperature of a black body producing the same luminosity as the star: $\mathrm{L{=}4\pi\ R^2\sigma\ T_{eff}}$ with $\mathrm{R{=}r(M)}$.

For the final missing boundary condition, we need to specify the outer pressure P(r=R) at R=r(M). This typically comes from imposing a *smooth* transition from the stellar interior (inside, $\mathrm{T \geq T_{eff}}$) and the stellar atmosphere (outside, $\mathrm{T \leq T_{edd}}$), which requires calculating the pressure in the stellar atmosphere where the assumptions we made so far, and consequently the equations we wrote do not hold.

- **Q**: Why should P be smooth? (**Hint:** think of dP/dr!)

Solving strategies

Let's assume we have a way to specify P(r=R), and postpone the discussion of stellar atmosphere to a future lecture. We then have 4 coupled non-linear ODEs with 4 boundary conditions (2 at the center and 2 at the surface), and the EOS as a closure condition. How can we solve them numerically?

Discretization

First, to represent the system of equations in a computer we want to discretize them, that is convert every derivative into a *finite difference*. This can be done in various ways, each with specific advantages and disadvantages. The simplest is a first-order forward discretization of the form:

$$\frac{df}{dx} \rightarrow \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} \quad , \tag{9}$$

where f and x are a generic function and variable, and the index k labels the discretized points.

For example, if x=m, the index k will label which cell of the mass-coordinate "mesh" we are considering and $\mathrm{m_{k+1}}$ - $\mathrm{m_k}$ = $\Delta\ \mathrm{m_k}$ is the local resolution in mass at location k, while if x=t, then k will label the "timestep" we are considering, and $\mathrm{t_{k+1}}$-$\mathrm{t_k}$ = $\Delta\ \mathrm{t_{k+1}}$ is the timestep size.

N.B.: typically both the spatial resolution $\Delta\ \mathrm{m_k}$ and the temporal resolution $\Delta\ \mathrm{t_k}$ are *adaptive*, meaning the stellar evolution code will put more mesh points where / take more timesteps when quantities vary more rapidly. This is necessary to deal with the large dynamic range of multiple quantities.

N.B.: Nature $\gg$ numerical models, because nature does not need to do this!

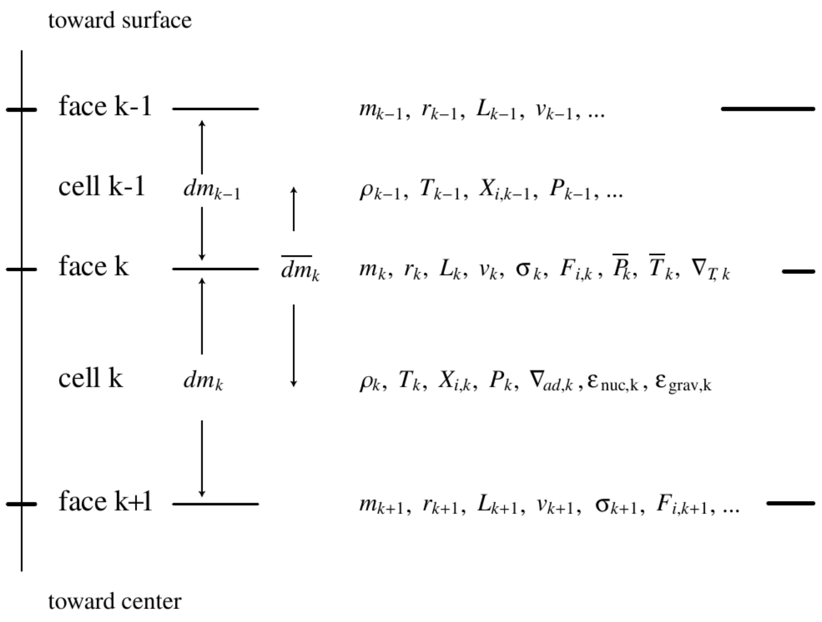
toward surface
face k-1
cell k-1
face k
cell k
face k+1
toward center
dm_{k-1}
$\overline{dm}_k$
dm_k
$m_{k-1},\ r_{k-1},\ L_{k-1},\ v_{k-1}, ...$
$\rho_{k-1},\ T_{k-1},\ X_{i,k-1},\ P_{k-1}, ...$
$m_k,\ r_k,\ L_k,\ v_k,\ \sigma_k,\ F_{i,k},\ \overline{P}_k,\ \overline{T}_k,\ \nabla_{T,k}$
$\rho_k,\ T_k,\ X_{i,k},\ P_k,\ \nabla_{ad,k},\ \varepsilon_{\rm nuc,k},\ \varepsilon_{\rm grav,k}$
$m_{k+1},\ r_{k+1},\ L_{k+1},\ v_{k+1},\ \sigma_{k+1},\ F_{i,k+1}, ...$

Figure 4: Schematic representation of the spatial mesh in MESA. Intensive quantities (T, ρ, P) that do not depend on the amount matter are defined at the cell "center", while extensive quantities (m, L) are defined at the cell "boundaries". This is Figure 9 in Paxton et al. 2011.

By discretizing the ODEs we can rewrite them as algebraic equations:

$$\frac{dm}{dr} = 4\pi r^2\rho \Leftrightarrow \ln(r_k) = \frac{1}{3}\ln\left(r_{k+1}^3 + \frac{3}{4\pi}\frac{dm_k}{\rho_k}\right) \tag{10}$$

$$\frac{dP}{dr} = -\frac{Gm(r)\rho}{r^2} \Leftrightarrow \frac{P_{k-1} - P_k}{0.5(dm_{k-1} - dm_k)} = -\frac{Gm_k}{4\pi r_k^4} \tag{11}$$

$$\frac{dT}{dr} = -\frac{3}{16\pi ac}\frac{\kappa\rho L}{r^2T^3} \text{ or } \left.\frac{dT}{dr}\right|_{\text{ad}} \Leftrightarrow \frac{T_{k-1} - T_k}{(dm_{k-1} - dm_k)/2} = -\nabla_{T,k}\left(\left.\frac{dP}{dm}\right|_k\right)\frac{\tilde{T}_k}{\tilde{P}_k} \tag{12}$$

$$\frac{dL}{dr} = 4\pi r^2\rho(\varepsilon_{\text{nuc}} - \varepsilon_\nu + \varepsilon_{\text{grav}}) \Leftrightarrow L_k - L_{k+1} = dm_k(\varepsilon_{\text{nuc}} - \varepsilon_\nu + \varepsilon_{\text{grav}}) \tag{13}$$

$$P \equiv P(\rho, \mu, T) \Leftrightarrow P \equiv P(\rho, \mu, T) \tag{14}$$

$$\left.\frac{dX_i}{dt}\right|_r = \left[\sum_j \mathcal{P}_{j,i}(T,\rho) - \sum_k \mathcal{D}_{i,k}(T,\rho)\right] + \left(D_i\nabla^2 X_i\right) \Updownarrow X_{i,k}(t_n + \Delta t_{n+1}) = X_{i,k}(t_n) + \Delta t_{n+1}\left(\frac{dX_{i,k}}{dt}\right) \tag{15}$$

Where, when going from the physical model to the numerical implementation:

- we use m rather than r as independent coordinate as expected;
- the mass continuity if re-formulated in terms of natural logarithm or r instead or r itself (to keep numbers smaller);
- the EOS, κ, $\varepsilon_{\text{nuc},}$, and ε_ν represent input physics that we borrow from other fields, and typically are *tabulated* as a function of T, ρ or equivalent pairs of variables.

Now that we have transformed a set of ODEs into a set of algebraic equations, the stellar structure and evolution problem is reduced to solving a matrix equation:

Writing this in a symbolic compact form we an array of quantities X(t) of length equal to the number of mesh points for our spatial discretization times the number of variables, and a matrix A which represents how all the

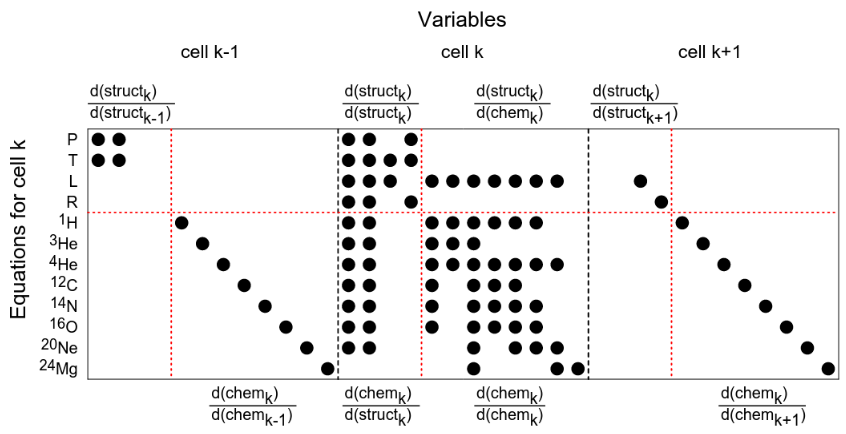

Figure 5: Section of an example of the matrix to solve. Black dots are non-zero entries, meaning the variables are coupled by an equation. Vertical dashed lines denote blocks of the matrix for cell `k-1`, `k`, and `k+1` respectively, red dotted lines separate structural variables and compositional variables. This is Fig. 47 of Paxton et al. 2013.

entries of X are coupled to each other in the algebraic form of the equations: $AX(t_k) = X(t_{k+1})$.

To solve this we could consider two approaches:

- **Explicit methods**: in this case $X(t_{k+1})$ is expressed as a function of $X(t_k)$. This however will require resolving the *local* dynamics (and be limited by the sound crossing time), making it impossible to perform a calculation for the *evolution* of the star
- **Implicit methods:** in this case we write a function $F(X(t_k), X(t_{k+1})) = 0$ (e.g., $F(X(t_k), X(t_{k+1})) \equiv AX(t_k) - X(t_{k+1}) = 0$), and by solving the homogeneous equation we have obtained we derive $X(t_{k+1})$. Implicit methods are preferred in stellar evolution as they are not limited by the sound crossing time.

Solver

For each of these methods, we still need to specify a numerical solver. Typically stellar evolution codes rely on generalized first order Newton Raphson solvers, which find the zeros of a function starting from an initial guess and then computing its derivative to create a better second guess:

https://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif

Shooting method vs. Henyey method

There is a peculiarity in the way we built the system of equations to be solved: we have two boundary conditions at one edge of the (one dimensional) domain (the center), and one set at the other edge (the surface).

Historically, the first computations of stellar models would use a "shooting method": integrate numerically from the center outward, and verify how far from the outer boundary conditions the solution lands, based on the distance, calculate a correction on the initial guess, and iterate. This was computationally expensive, and would require many iterations before the solution build "inside out" would be sufficiently good.

In the late 1960s, L. Henyey designed an alternative solving strategy, which was rapidly adopted by others and is still widely used today. The key idea was to not start computing the variables inside out, but instead, make an initial guess for the entire domain (typically the initial guess can be the solution for the previous timestep!), and calculate a correction not based on the distance between the solution and the outer boundary condition only, but based on both boundaries at the same time!

N.B.: for a biased overview of the evolution of the field of computational stellar physics by one of the "founding fathers" of the field, see this 1978 interview to R. Kippenhahn.

Evolution splitting

One final peculiarity of the stellar structure and evolution problem is that the evolution is *slow* and driven by nuclear physics, which provides some of the hardest equations to solve, because they are extremely *stiff*. Mathematically, this means the eigenvalues of the nuclear physics part of the problem have very different norms, making it challenging to design numerical algorithm that can find all the eigenvalues efficiently. Physically, this is because many reactions have extreme temperature sensitivities (because of the Coulomb barriers and the tunneling probability).

The clear separation of timescales for the structure and evolution ($\tau_{\mathrm{ff}} \ll \tau_{\mathrm{KH}} \ll \tau_{\mathrm{nuc}}$) allows for one more trick, so called "evolution splitting": one can compute the *structure* of the star assuming fixed composition, then *evolve* the composition with that fixed structure (meaning T, and ρ), and iterate computing a new structure with the updated composition. This allows to separate the problem and can make it more tractable.

Open science: MESA and `MESA-web`

"*An algorithm must be seen to be believed*" - D. Knuth

The discussion above is an attempt to highlight the general principles and technical aspects specific of computational stellar structure and evolution without narrowing down to a specific code. Historically, many groups independently developed computer codes to solve the stellar structure and evolution problem, and only few universities would have the computing power to run them. In the 1960s-1980s, even if the group leaders may have been willing to share their codes upon request, all the algorithmic details would typically be hidden from the scientific community, and considered as technicalities. Many of the people actually *writing* the code and *implementing the algorithms* were women, and their contribution has been under-appreciated because of this attitude of considering algorithms as technicalities.

Still today, the algorithmic details of many codes are only known to the inner circle of the group owning the code. In 2011, an independently wealthy computer-scientist-turned-astrophysicist free from the publish-or-perish academic mentality named Bill Paxton, with the collaboration of a large group of theoretical and computational astrophysicists (notably, P. Eggleton who provided his code as a starting point), developed a new open-source, multi-purpose, and community driven stellar structure and evolution code, called `MESA`. Because this is the tool I use, and the only one I can really inspect, a lot of my intuition on the numerical methods and computational aspects is tied to this particular code. The `MESA-web` interface you have been using is just a predefined setup of this very flexible and modular code designed for educational applications: you have been using a bleeding edge research tools!

The release of this code, together with a change in perspective from funding agencies and a general push towards open-science and open-know-how (which necessarily require to get a handle on the algorithmic details), has significantly changed the field of theoretical and computational stellar evolution. Other codes ("the GENEVA code" - a.k.a. GENEC, "the Frascati code" - a.k.a. FRANEC, "the Cambridge code", KEPLER, HOSHI, etc.) are progressively opening up, and at the very least being forced to compare their results with MESA.

N.B.: the fact that most codes don't even have a name and are identified with the institution where they are mainly developed is an indication of the secrecy behind the computational aspects: you had to be at the right place to get a handle on all the details!

This openness allows finally to check *systematic modeling errors* that arise from algorithmic and numerical choices that may differ even for the same physical assumptions and input physics. This has enabled progress, revealed (always existing) bugs in MESA and in other codes, and overall improved the state of this field.

N.B.: Multiple independent codes are needed to cross-check and validate results. The exponential growth of the community of users of the MESA code has almost become a problem by quenching the use of other codes in this problem space!

Summary

1. take a star, which is a complex multi-scale, multi-physics, multi-dimensional object

2. obtain from other fields of physics input for $\varepsilon_{\mathrm{nuc}}$, ε_ν, κ, and the EOS

3. write down an approximate physical model assuming spherical symmetry, going from a multi-dimensional object to a continuous line along a spatial direction

4. discretize that line into "chunks" of (variable) size $\Delta\ \mathrm{m_k}$

5. use an implicit first-order generalized Newton-Raphson solver to determine how variables at the points $\Delta\ \mathrm{m_k}$ vary within discrete (variable) time interval $\Delta\ \mathrm{t_k}$.

6. **make sure that the scientific results you obtain do not depend on any of steps 2-5**.

Population synthesis

As we saw since the lecture on the CMD/HRD, in stellar physics it is often necessary to rely on *population studies* to infer something about the physics of the star. Moreover, when thinking about galaxies (e.g., integrated light from a far away galaxy where not all stars are resolved), we definitely need *populations.* Population studies are also needed to predict *rates* of phenomena (e.g., supernovae, gravitational wave mergers, etc.).

The practice of *combining* many stellar models to make a population is generally referred to as *population synthesis.* This can be done with *detailed* models, such as the one coming from codes that solve the equations of stellar structure (e.g., MESA, KEPLER, FRANEC, PARSEC, HOSHI, etc.).

N.B.: These codes being 1D typically are *relatively* cheap to run, about ~1-10× CPUh per model.

However, often there are many things one may want to change for *fixed* stellar evolution (e.g., the initial distribution of stars, or their metallicity distribution, or how they interact in binaries or higher multiplicity systems), and recomputing all the models is not necessary.

Since the 1990s, there has been the development of "semi-analytic" techniques based on implementing polynomial fits to a set of stellar models (e.g., Pols et al. 1998), and use these to simulate the evolution of stars - producing a *rapid* population synthesis. Coupling these to analytic models for the interaction of binaries (which are used by detailed stellar codes too!), one can get *rapid* population synthesis.

N.B.: In these codes the stellar structure problem is "pre-solved" and the use of semi-analytic fitting formulae to those solutions saves a lot of time, going from 10s of CPUh per star/binary to ∼100 CPU-millisecond per star/binary.

More recently, there has been a move to use directly tabulated results from detailed stellar evolution codes, instead of analytic fits to those tables, which allows to track more details (and more easily update the stellar models) for a small computational price that is now affordable.

Below is an (incomplete, biased) list of some of these codes.

Semi-analytic approach to population synthesis

- BSE: the forefather of many, see Hurley 2002 and references therein (**N.B.:** not *all* rapid population synthesis codes)
- COSMIC: Open source and open development, `pip` installable
- COMPAS: Open source
- `binary_c`: Open source, python front-end available, can be run on online (similar to `MESA-web`)
- ...

Pre-computed and tabulated or hybrid codes

- POSYDON (`MESA` tables of models, open source)

- BPASS (Based on models from "the Cambridge code")
- ComBiNe (`BEC` tables of models)
- METISSE (`MESA` tables of models, open source)
- MINT (= `binary_c` + `MESA` tables of models, not yet public)
- SEVN (based on `PARSEC/FRANEC` tables of models, open source)
- ...

Homework: preparation for in class activity

In two weeks from now we are going to do a class activity to piece together all the things we have learned so far and describe the evolution of single, non-rotating stars with *your* `MESA-web`-produced models! Although we will stay close to the textbook stellar evolution, this activity is similar to actually doing research with stellar models: you compute a model, and see if it makes physical sense based on physical intuition and contextual knowledge – keeping in mind that models can and do produce numerical nonsense occasionally, and one has to be careful to not fool oneself! Typically in research I will ran further models changing the input to test my interpretation, run numerical tests (change the spatial and temporal resolution for example), and iterate.

This preparation homework is not graded, I just ask you to gather these models that we will discuss in class. Each student will have one (randomly assigned) mass to run with MESA web, you will need to download and unzip on your laptop and have it ready by *<2024-10-22 Tue>*.

In class, you will gather in groups based on the masses you have run, and be discuss among yourselves why the evolution proceeds the way it does, whether you think it is physical or numerical artifacts, and we will then discuss some of these models with the whole class.

We will mostly used the "movies" that `MESA-web` produces for you, but feel free to prepare to read profile files and/or history files in a script and plot other things to better understand what goes on.

N.B.: The `MESA-web` interface can get clogged if you all ask calculations the night before, so please to this well ahead - it's just filling an online form!

N.B.: for the sake of having a uniform set of models to discuss, please use the options listed below for your `MESA-web` submission (I listed all options but the mass and highlighted in **bold** the ones to change w.r.t. the defaults).

N.B.: if the models below do not reach the desired stopping condition, feel free to try again with different parameters (especially "Variance Control Target" and "Mesh Delta Coefficient" which control the temporal and spatial resolution, lower means higher resolution for both). Please keep the first model around too for the in class activity.

Initial mass $M \leq 7M_\odot$

- **Nuclear Reaction Network**: `approx21`

Stopping condition $\log(\rho_{\mathrm{center}}/([\mathrm{g\ cm^{-3}}])) \geq 10$

Not all models may successfully reach this condition. We will have a chance to discuss "failed" models!

Initial mass $M > 7M_\odot$

- **Nuclear Reaction Network**: `approx21`
- **Semi-Convection Alpha**: 0
- **Thermohaline Alpha**: 0
- **Red Giant Branch Wind Scheme**: Dutch (this specifies stellar winds)
- **RGB Wind Scaling Factor**: 0.8
- **Asymptotic Giant Branch Wind Scheme**: Dutch
- **AGB Wind Scaling Factor**: 0.8

Stopping condition: iron core collapse (default) value `1e8`

The value specified the speed of collapse that the iron core needs to reach before the model stops. Usually by this point of the evolution ρ is so high that ν interactions start to matter, the EOS is not that of an ideal quantum gas + radiation, and the time left before a NS is formed is of order of $\sim$100 milliseconds (see also lecture on supernovae).

Not all models may successfully reach this condition. We will have a chance to discuss "failed" models!