Appendix D

Modelling a stellar core

Term project 1[1]

The first term project involves modeling the central parts of the sun by writing a computer program to solve the governing equations. The project involves information contained in Chap. 1–5.2.1.

Solve the equations governing the internal structure of the radiative core of the Sun. You are free to use the programming language you prefer, and the numerical method you prefer. The equations you are to solve are the following:

$$\begin{aligned}
\frac{\partial r}{\partial m} &= \frac{1}{4\pi r^2 \rho} \\
\frac{\partial P}{\partial m} &= -\frac{Gm}{4\pi r^4} \\
\frac{\partial L}{\partial m} &= \varepsilon \\
P &= P_G + P_{rad} \\
\frac{\partial T}{\partial m} &= -\frac{3\kappa L}{256\pi^2 \sigma r^4 T^3}
\end{aligned}$$

The first differential equation comes from the expression for mass within a radius $m(r)$. The second equation comes from the assumption of hydrostatic equilibrium. The third comes from the definitions of L and ε. The final differential equation is a solution to the radiative diffusion equation (with no conduction or convection).

[1] This term project has been improved by Odd Petter Sand (2015)

The reason m is used as the independent variable is that the set of equations is more stable than it would be if we used r (but we will still use the r results to make plots once we are done with solving the equations).

You will notice that you have 7 unknowns $(r, \rho, P, L, P_G, P_{rad}, T)$ but only 5 equations (we assume m, κ and ε to be known, since we choose m, pick κ from a table (see Tasks) and can adapt the calculations from App. C to get ε). Thus, you need two more equations. You can assume an ideal gas to get an equation of state, and you'll need an equation for the radiation pressure as well. All the symbols have the standard meaning.

NOTE: There is no equation for $\frac{\partial \rho}{\partial m}$; this is by design. Solving such an equation in your program will lead to unphysical results.

Assumptions

- All the assumptions from App. C.
- Ideal gas.

Goals

Your final code should produce a star that:

- Has L, m, r all going to 0 or at least within 5% of L_0, M_0, R_0, respectively.
- Has a core ($L < 0.995$) reaching out to at least 10% of R_0.

Initial parameters

As a starting point, we input the actual values at the bottom of the solar convection zone (except P, which will differ because we are assuming an ideal gas, and M, which is simply the mass of the entire Sun). Note that $\bar{\rho}_\odot$ is the average density of the Sun: $\bar{\rho}_\odot = 1.408 \cdot 10^3 \,\mathrm{kg\,m^{-3}}$

$$
\begin{aligned}
L_0 &= 1.0 \cdot L_\odot \\
R_0 &= 0.72 \cdot R_\odot \\
M_0 &= 0.8 \cdot M_\odot \\
\rho_0 &= 5.1\bar{\rho}_\odot \\
T_0 &= 5.7 \cdot 10^6\,\mathrm{K} \\
X &= 0.7 \\
Y &= 0.29 \\
Y_3 &= 10^{-10} \\
Z &= 0.01 \\
Z_{{}^7_3\mathrm{Li}} &= 10^{-13} \\
Z_{{}^7_4\mathrm{Be}} &= 10^{-13}
\end{aligned}
$$

For the final task (see Tasks), it should be possible to find a solution with no parameter changing more than a factor of 5 (i.e. from $\frac{1}{5}$ to 5 times the value in the table). Yes, that means you can indeed increase R_0 so it is larger than the radius of the Sun.

You may not change the mass fractions X, Y, Z, the luminosty L_0 and the mass M_0, only the other parameters.

Tasks

- Create a method that reads the file the file **opacity.txt**[2] (found on the course webpage) and takes T and ρ (in SI units) as input and returns κ (in SI units). You will need to use linear 2D interpolation for the common case where the input value is not exactly found in the table. Your code should also be able to extrapolate if you outside the bounds of the table (though you may want to have your program output a warning when you do so). Test your methods against the first sanity check below and see that you are able to reproduce the results shown there. The structure of the file is as follows:
  - The top row is $\log_{10}(R)$, where $R \equiv \frac{\rho}{\left(\frac{T}{10^6}\right)^3}$ and ρ is given in cgs units ($\frac{\mathrm{g}}{\mathrm{cm}^3}$).

[2]The opacity table is taken from Iglesias & Rogers, 1996, ApJ, vol 464, p. 943

  - The first colum is $\log_{10}(T)$, with T given in K as usual.
  - The rest of the table is $\log_{10}(\kappa)$, again in cgs units ($\frac{\mathrm{cm}^2}{\mathrm{g}}$).

- Implement methods to calculate $\rho(P,T)$ and $P(\rho,T)$. (Why not also for T?)
- You are now ready to begin on the main body of the program: Solving the four partial differential equations. You can choose which ever numerical method you prefer (Euler, Runge-Kutta, Simpson's method, for instance). It is extremely helpful to print the parameters to screen for every 50 steps or so.
- Have your program output final values of the parameters.
- Implement dynamic step size (see the file **variablesteplength.pdf** on the course web page for details). Make sure you can turn dynamic step size on and off as needed.
- See "Sanity checks" and "Report" below and have your program create the plots you will need.

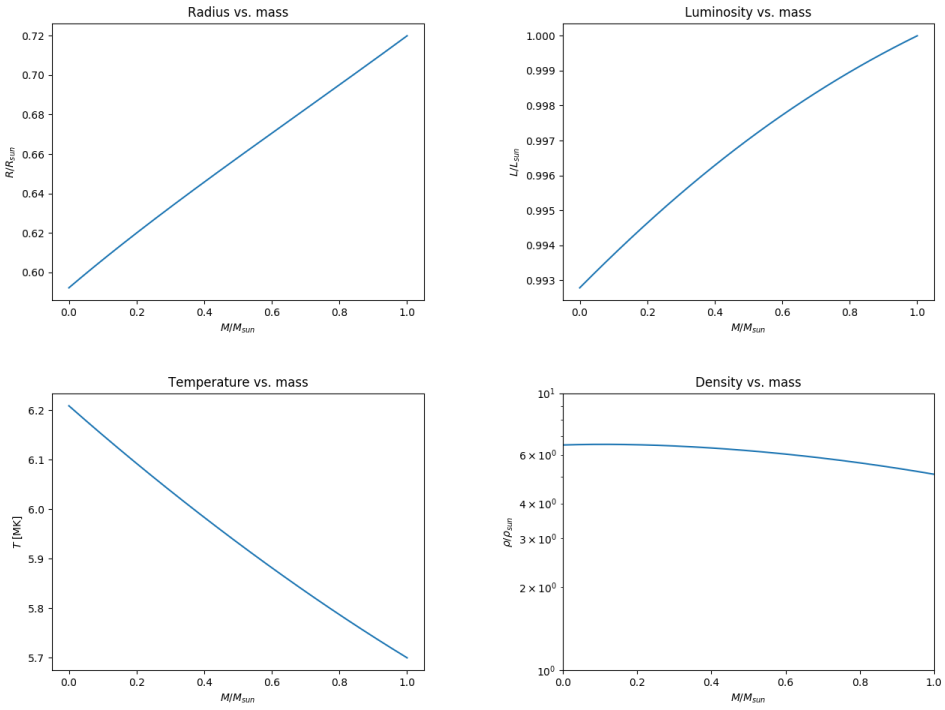
Sanity checks

Your first sanity check is to test your functions $\rho(P,T)$ and $P(\rho,T)$. Check that $\rho(P(\rho_0,T_0),T_0)=\rho_0$ and similarly for $P(\dots)$ so you're confident that your equation of state functions work both ways.

Interpolation of opacity values should give these results (or something very close to them):

$\log_{10} T$	$\log_{10} R$ (cgs)	$\log_{10} \kappa$ (cgs)	κ (SI)
3.750	-6.00	-1.55	$2.84 \cdot 10^{-3}$
3.755	-5.95	-1.51	$3.11 \cdot 10^{-3}$
3.755	-5.80	-1.57	$2.68 \cdot 10^{-3}$
3.755	-5.70	-1.61	$2.46 \cdot 10^{-3}$
3.755	-5.55	-1.67	$2.12 \cdot 10^{-3}$
3.770	-5.95	-1.33	$4.70 \cdot 10^{-3}$
3.780	-5.95	-1.20	$6.25 \cdot 10^{-3}$
3.795	-5.95	-1.02	$9.45 \cdot 10^{-3}$
3.770	-5.80	-1.39	$4.05 \cdot 10^{-3}$
3.775	-5.75	-1.35	$4.43 \cdot 10^{-3}$
3.780	-5.70	-1.31	$4.94 \cdot 10^{-3}$
3.795	-5.55	-1.16	$6.89 \cdot 10^{-3}$
3.800	-5.50	-1.11	$7.69 \cdot 10^{-3}$

If your code works correctly, you should get (approximately) the following plots using the given initial parameters and a step length of $-10^{-4}\mathrm{M}_{\odot}$:



Some notes on debugging your code

For some of you, this project represents a significant step up in scope and complexity from what you are use to in programming projects. As such, here are a few pointers for the all but inevitable situation that something is wrong with your code, but you don't know what. **Read this before you start writing your code,** and keep it in mind throughout the project:

- Don't write the entire program all at once. Take one task at a time and use the sanity checks whenever available. When you're convinced it works as it should, you can move on to the next task.

- Repeat the process above for tasks you've already completed if you can't find a bug. It's possible you changed something so that your old code no longer passes the sanity checks.

- Only fix one bug at a time.

- If you get bugs that crash your program (giving an error message or doesn't compile[3]), read carefully what the error message says. If you don't know what it means, copy the error message into a search engine. There's almost always someone else out there that has encountered the same thing before you.

- Start from the top of your code, and fix those errors first. A later error might have been caused by the earlier one - make sure that this is not the case before moving on to the next bug.

- Print out all your variables for every step if you can't locate the error, and see if anything's off. In Python, use **print** statements to output the values of variables you want to check on. You may use **input()** to pause your program so you can look at the printouts before continuing. If not using Python, you can look up how to do this in your language of choice. (If you have access to a proper debugger that can set break points (i.e. pause the program, so you can execute one line of code at the time) and watches (watch how the value of a variable changes for each step), that's even better, but entirely optional.)

- Common causes of bugs that are notoriously hard to track down:

  - Long division. Every time you're working with real numbers, make sure to use real literals like 1.0. and 2.0 - do not use integer literals like 1 and 2 - these are dangerous. Look at the difference between **print exp(1/2)** and **print exp(1.0/2.0)** and recoil in horror as you contemplate what this would do to your results.
  - Avoid variables with the same name at all costs. In Python, you cannot flag a variable as constant (as in "forbidden to overwrite once declared"), so it is possible to overwrite a constant if you're not careful. Blame the Python creators for this unfortunate possibility. Example: Defining the speed of light $c = 2.998 \cdot 10^8$ and then later using an integer counter named c is a typical example of a bug which can be very difficult to track down: "Wait, how did I end up with a light speed of zero?"
  - (Add your own and suggest them to the teacher for the benefit of future students.)

[3] Only applies if you're not using Python. Some other languages require you to compile your program before running it.

The report

You are required to write a report in this exercise - writing down the method you use and in which order you solve the equations, etc. You also need to include what problems and what you have found out while solving this exercise, this includes both numerical problems, resolution problems and problems with the physics. The report needs to be well structured and understandable. The way your report is written does impact the number of points you will get for this project.

The report has to include a file with your code so that it can be run on a non-MS Windows machine and any licenses required must be available at the Institute of Theoretical Astrophysics. Your code will have to be written so that not only yourself can read it, and it will have to be commented, meaning that when defining a variable or constant, it has to be explained as well as when calculating something that will have to explained as well. The code should be structured logically so that it is easily understandable.This is a very important point - if I do not understand what happens in the code, it will mean a serious deduction in the number of points you will get - even if you successfully solve it!

There will be no extensions to the deadline, except in case of documented medical circumstances. If you cannot solve it, write the report anyway, explaining your problems and what you tried in order to solve them.

Your report should contain the following:

- Describe how you calculate the average molecular weight, μ.
- Plot r as a function of m with dynamic step size turned off for significantly different step sizes ∂m. Once you have sufficient data, comment on the effects of changing the step size.
- By running several experiments, you should report what happens when you
  - change the value of R_0 (keeping all other initial parameters unchanged).
  - change the value of T_0 (keeping all other initial parameters unchanged).
  - Change the value of ρ_0 (keeping all other initial parameters unchanged).
  - change the value of P_0 (keeping all other initial parameters unchanged).

You need to run several experiments for each of these – not just two or three.

  - Explain why the changes happen.
  - Does changing some of the parameters produce very similar effects?
  - Are there cases where you may compensate for changing one parameter by changing another (and if so, in which direction should you change them)?

  There is no limit on how many words you should use, but make it as brief as you can without leaving out anything important. A bulleted list is fine for this purpose if that is what you prefer.

- Finally, try to create a model where both the radius and the luminosity becomes zero when the mass becomes zero (See "Goals"). You are here free to change any of the three initial parameters we changed in the previous task (ρ_0, P_0, T_0 and R_0).

  - A good starting point is trying to make m and r go to 0 (+/- 5% of the initial value) at the same time, using what you've learned so far.
  - Pay special attention to what happens to ρ as a consequence of how r behaves. Study them and see what happens. Comment on the results.
  - It's easier to get convergence to 0 if your core is very small (L drops very fast). In other words, there is a tradeoff to be made here: By having one of your variables not go all the way to 0, you can get a more realistic core.
  - Extrapolating in the opacity table is most likely inevitable. Don't worry about this if it happens.

- For your best model, plot the values of T, L, ε, ρ, P and m, all as a function of the radius r (to get an idea what a cross section of your star actually looks like). Use a logarithmic axis for ε, ρ and P. Explain your choice of parameters by using these plots.

- The report can be no longer than 10 pages.