Astro 220A Quarter Project

Due Friday, December 5, 2014

Basic Instructions for the Calculation of a Stellar Model

(The original famous assignment from Peter Bodenheimer)

1) Pick a value for the mass and composition (X, Y, Z). Recall that X (H mass fraction) + Y (He mass fraction) + Z (metals mass fraction) = 1.0

2) Find an appropriate opacity table. Write your own interpolation routine to find the opacity at a given temperature and density.
   There are at least three sources for data. The first is OPAL:
   http://opalopacity.llnl.gov/
   A variety of already computed tables can be downloaded, and there is a Fortran routine that you can use to help you in designing your program. It can be exceedingly similar to the provided Fortran program, if you choose. You can also have OPAL compute opacities for you, for any mixture. Probably you want "Type 1" tables, which have "standard" abundance ratios for the various metals
   The second is the Opacity Project (OP):
   http://cdsweb.u-strasbg.fr/topbase/TheOP.html
   OP allows you to compute opacities, for any mixture, at the OPServer:
   http://opacities.osc.edu/rmos.shtml
   Tables are in the OPAL format, or they can be customized.

   Note that the OPAL/OP tables only reach down to log T = 3.75 (or perhaps 3.50). *This may affect your choice of model star mass.* For cooler temperatures see:
   http://webs.wichita.edu/physics/opacity/ Their tabulations are also in the OPAL format.

3) Write you own simple routines for calculation of the rate of energy generation by hydrogen burning, given density, temperature, and composition, and for the equation of state, where you will need the density as a function of pressure, temperature, and composition.

4) Set up the inner and outer boundary conditions.

5) Read sections 17.0, 17.1, and 17.2 of the "old" Numerical Recipes (Press et al., 1992, 2nd edition). The PDF version of the book can be found online various places, including here:
   http://www.nr.com/oldverswitcher.html (All versions)
   We will be using the method of shooting to a fitting point (section 17.2). You can use the routine given there (subroutine shootf) to perform one integration. You will need an overall control program (something like newt, Section 9.7) to do multiple calls to shootf, to calculate corrections to the assumed boundary values, and to eventually obtain a converged solution. Modify the programs for your own purposes. For example, it is generally helpful to take only a fraction of the correction to the unknown parameters that the program calculates.

   We have these Numerical Recipes routines available. If you login to one of the public machines like mambo, and go to:
   /opt/share/recipes/ you will Numerical Recipes routines find in Fortran or c:
   shootf, odeint, lubksb, ludcmp
   In Fortran, rk4 and rkqc are also needed, or in c, rkck and rkqs. If you chose to use IDL, a number of these routines are already included. I have no idea about Python, Ruby, etc.

6) The write-up is as important as obtaining a good solution. A good write-up should include:
   a) Statement of parameters used
   b) Statement of the physics problem
   c) Statement of the numerical method and how well it converged
   d) Tabular presentation of results
   e) Graphical presentation of results
   f) Complete listing of your computer program (Python, c, IDL, etc.)
   g) Comparison of your model with a more detailed model, such as one published in the literature. Try here, perhaps: http://www.astro.wisc.edu/~townsend/static.php?ref=ez-web
   h) Conclusion

"To err is human, but to really foul things up requires a computer."

The Advice Section:

1) IDL users: There may be an issue with double precision in IDL not being precise enough, depending on the platform, for the newton-raphson matrix inversions. However, in the past the built-in IDL NEWTON routine has worked for people. If you write your own newton-raphson routine in IDL you may run into problems, or the code may only work on some platforms. This has not been a problem with other languages, as far as I know.

A voice from the past:
"What I found was that my code worked on some machines, but not on others. This was probably due to the presence/absence of extended double precision. It works on my Mac, but not on all Linux boxes."

2) "I would say Newton-Raphson is extremely finicky (not just in its IDL incarnation, but in general). Nothing will defeat it more effectively than bad guesses for the star's central pressure and temperature. You don't even have to be way off, just slightly off. I bet some of the students have codes that work, but they haven't found the right initial conditions."

"I recall that the convergence was quite finicky and my initial 'guesses' were fine-tuned by hand to be numbers that did not loop infinitely, but would allow for convergence."

3) Your fitting point can be a mass shell at m=0.5M (or whatever), but it doesn't have to be. I could be 0.2M, if that helps.

4) A few bits that are somewhat related:
i) The increments used to get the numerical derivatives used in the matrix should not be too small (say 0.1% of the variable being incremented)
ii) Once the corrections are derived, they should not be used 'full force.' It sometimes means taking only a very small fraction of the calculated corrections to get the convergence process started. Maybe as small as 1%.
iii) Be careful of the step size used in the numerical integrations of the ODE's. For the inward integrations the step size (in mass) has to start out being very small. At the center this is not a problem.
iv) Particularly for the surface to core integration, adaptive step sizes can significantly speed up execution speed, although they are not required. Using an adaptive step size allows one to resolve the steep temperature gradient near the surface while still taking relatively large mass steps in the interior.

5) Length of the stellar model writeup: Not including figures, something like 5 single-space pages should be fine. Longer is also fine, but not TOO much longer.