Title: March 16, 2012 Research/Programming Notes & Progress Copy Copy
Date: March 16, 2012 4:26 PM
Category: Work
Tags: python, research, Henyey code, Bodenheimer code, finding initial conditions, from campus

March 16, 2012 9:06 PM

Location: from home
Computing context: Macho-Mac2

Helpful computer hints discovered today:

- To open an application (e.g. Preview) from the command line, type:
  - open -a preview.app <path to file you want the app to open>
- To find the full path to where your python modules (that you, say, installed from the internet) are actually living on your filesystem, type the following command in at the python prompt:
  - <moduleName>.__file__

Continuing from last time:

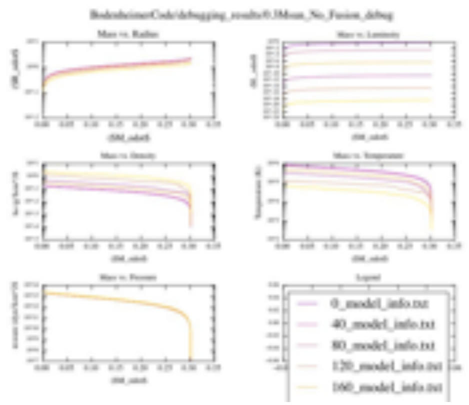
- **Actual science stuff w/ the code:**
  - ~~Plot the 0.5Msun (w/ no fusion) run results~~
    - ~~First, need to re-run the 0.5Msun dTval_12 simulation, but make sure to set it to output iteration info along its way to convergence. Then, plot that.~~
  - Done.
  - ~~How to incorporate running that sed command on the models' headers (if necessary) into my python code?~~
  - **Done. See the stuff near the start of** /Users/laurel/Desktop/Research/BodenheimerCode/MyPythonGUIPlottingScript.py
    - for the code/commands that accomplishes this.
  - ~~Also need to figure out how to make the legend smaller, b/c it's running off the final graph~~
    - Done:
    - In /Users/laurel/Desktop/Research/BodenheimerCode/MyPythonGUIPlottingScript.py :

```
params = {'legend.linewidth': 0.25}
rcParams.update(params)
legend(loc=2,prop={'size':8})
```

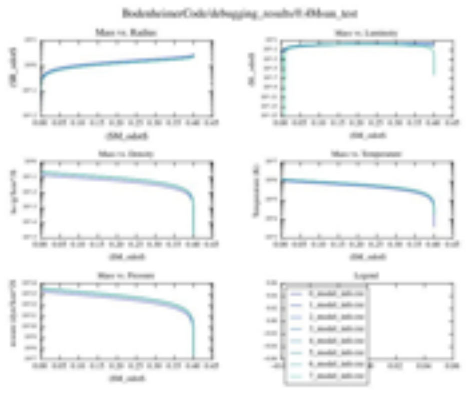
  - ~~Try to answer "why aren't the under 0.5M$_{\odot}$ (w/ no fusion)~~

~~simulations converging, regardless of their timestep sizes?”~~

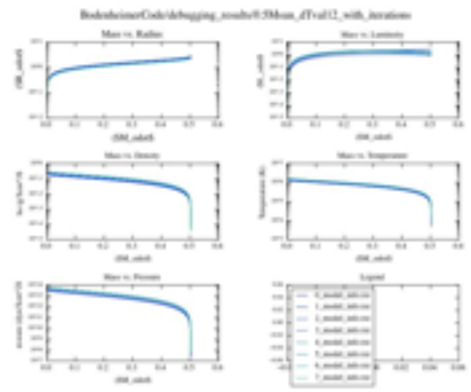
- ~~Compare them to the 0.3Msun (no fusion) results.~~
    - ~~Try to figure out why the 0.3M$_{\odot}$ is going off the rails, but the 0.5M$_{\odot}$ isn’t.~~
- ~~Run thecode.f with 0.40M$_{\odot}$ (no fusion)~~
    - ~~plot the results~~
    - ~~again, try to spot why it might be going off the rails~~
- Well, I've compared them... Here's what they look like:
    - The 0.3Msun case (does not converge):



    - The 0.4Msun case (does not converge):



    - The 0.5Msun case (which converges):



- I'm not sure what to make of this. The lower mass/unconverged ones are clearly showing the 'oscillating away from a stable solution' behavior, but I don't know why. It's possible that the lower mass models can't converge **from a polytropic initial condition** right off the bat without fusion.
- So! Try running the initial (polytrope-->model) single timestep for these lower mass models **with** fusion turned on. Then, see if you can take that initially converged model (which is no longer in any way, shape or form the initial condition that so set these models off the rails) and successfully evolve it forward in time **without fusion**
- Steps:
  (1) For the 0.3Msun case:
    (a) Run polytr.f on polytr03.inp (saving the result as polyout)
    (b) Edit pmsstar03.start, setting
      - The 'write to file' filename to... 0.3MsunWithFusion.mod. Or something like that.
      - EFRC = 1.0 (i.e., turning fusion back on)
      - NTES = 5 (if it isn't already, just so we can compare the convergence behavior of the successful case to the ones above)
      - dTIM (and everything else on that line) to 1.00E+11
    (c) Run thecode.f on pmsstar03.start. If it fails to converge, try using a smaller timestep. Don't go any father along this checklist until you get it set up

so that it *does* converge.

   (d) Run ./parse_output.pl on the (ascii) results. Also, copy over the .mod file to the results directory.
   (e) Edit pmsstar03.start, setting
      - The 'input file' to binary, name = 0.3MsunWithFusion.mod
      - The 'write to file' filename to... 0.3MsunNoFusion.mod.
      - EFRC = 0.0 (i.e., turning fusion off)
      - NTES = 5 (if it isn't already, just so we can compare the convergence behavior of the successful case to the ones above)
      - dTIM (and everything else on that line) to 1.00E+11
   (f) Run thecode.f on pmsstar03.start.
   (g) See if the model still fails to converge.

(2) For the 0.4Msun case:
   (a) Run polytr.f on polytr04.inp (saving the result as polyout)
   (b) Edit pmsstar04.start, setting
      - The 'write to file' filename to... 0.4MsunWithFusion.mod. Or something like that.
      - EFRC = 1.0 (i.e., turning fusion back on)
      - NTES = 5 (if it isn't already, just so we can compare the convergence behavior of the successful case to the ones above)
      - dTIM (and everything else on that line) to 1.00E+11
   (c) Run thecode.f on pmsstar03.start. If it fails to converge, try using a smaller timestep. Don't go any father along this checklist until you get it set up so that it *does* converge.
   (d) Run ./parse_output.pl on the (ascii) results. Also, copy over the .mod file to the results directory.
   (e) Edit pmsstar04.start, setting
      - The 'input file' to binary, name = 0.4MsunWithFusion.mod
      - The 'write to file' filename to...

0.4MsunNoFusion.mod.
- EFRC = 0.0 (i.e., turning fusion off)
- NTES = 5 (if it isn't already, just so we can compare the convergence behavior of the successful case to the ones above)
- dTIM (and everything else on that line) to 1.00E+11

(f) Run thecode.f on pmsstar04.start.

(g) See if the model still fails to converge.

- See if using the ‘mass chain-down’ technique with the 0.5M$_{\odot}$ (no fusion) converged model as a starting point can produce converged models for lower mass (no fusion) balls of gas.
    - Implement a mass chain-down proceedure in thecode.f
        - Add a ‘mass chaindown?’ flag to the .start file, and modify thecode.f to be able to read it in
        - If the ‘mass chaindown’ = true,
            - read in a converged model
            - evolve it forward in time by 10(?) dTthresh steps
            - then decrease the mass of the system by some factor
                - (By what factor? How much or how little can you successfully decrease the mass at any given chain-down step? Need to think about this more once I get to this point...)