Title: March 15, 2012 Research/Programming Notes & Progress
Date: March 15, 2012 1:29 PM
Category: Work
Tags: python, Bodenheimer code, research, finding initial conditions, from campus, Henyey code

March 15, 2012 1:39 PM

Location: on campus
Computing context: Macho-Mac2

From last time:

- **Coding housekeeping stuff:**

1. ~~Modify the python plotting script to take a directory name as an input argument~~
   - ~~Done. It checks the last argument in the argv list, and if it's not an existing directory, prompts the user to enter another file path. (And keeps prompting the user thus until s/he enters the name+path of a directory that *actually exists*~~
2. ~~And possibly also a basename for 'the type of file we should be looking to plot'~~
   - ~~Come back to this later.~~
3. ~~It'd be great if I could write a python script that did the model #/ iteration #/ corrections #/ evolution/ dTthresh plotting stuff all from the same script...~~
   - ~~Come back to this later.~~

Accomplished (2) and (3-ish), and eliminated the purpose for (1) by adding in GUI front-end to my python general-purpose plotting code, which is now living in /Users/laurel/Desktop/Research/BodenheimerCode/MyPythonGUIPlottingScript.py

Thank you, easygui package! Tutorials, download, etc. available from here: http://easygui.sourceforge.net/index.html

4 Incorporate that header-fixing sed command into the body of the python code... somehow.... how?

The code that does this (included here in script form) more-or-less is:

```
#-----------------------------------------------------------------------------------------------
#! /Library/Frameworks/Python.framework/Versions/Current/bin
```

```
import os, sys, re

currentdir = os.getcwd()
files = os.listdir(currentdir)

for name in files:
    if ".txt" in name:
        with open(name,"r") as source:
            data = source.read()
            if ( re.match('J\s+dM',data)):
                changed = data.replace("J", "J      c")
                #               print "Goodbye, old header!"
                #               print name
                with open(name,"w") as target:
                    target.write(changed)

#-----------------------------------------------------------------------------------------------
```

Now

- **Actual science stuff w/ the code:**
  - Try to answer “why aren’t the under 0.5M$_{\odot}$ (w/ no fusion) simulations converging, regardless of their timestep sizes?”
    - Plot the 0.5Msun (w/ no fusion) run results
      - How to incorporate running that sed command on the models’ headers (if necessary) into my python code?
    - 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 0.45M$_{\odot}$ (no fusion)
      - plot the results
      - again, try to spot why it might be going off the rails
  - 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...)