Laboratory Project:
Solar Oscillation Spectrum Analysis

Prof. Ilídio Lopes
José Vargas Lopes
(josevlopes@ist.utl.pt)
Astrophysics Laboratory

October 2, 2018

PART I: Modelling the Sun

The objective in the first part of the Astrophysics Laboratory is to use a stellar evolution code to simulate a realistic solar model, and obtain its acoustic modes of vibration. You will use the well estabilished *MESA* code, which computes a one dimensional quasi-hydrostatic stellar evolution, and *GYRE*, an adiabatic oscillation package to compute the oscillation frequencies. Both the packages are installed in the laboratory computers. If you want to install *MESA* and *GYRE* in your personal computer follow the steps in `http://mesa.sourceforge.net`.

1. Familiarize yourself with basic terminal commands and *MESA* usage by creating a simple star model using the default input physics.

   (a) Create a new default Working Directory (WD) by copying the directory `$MESA_DIR/star/work` to your work environment. In this folder you will find all the necessary files to run a stellar model[1].

   (b) The `inlist*` files in the WD are used to control the input physics for the model. The inlists found in your WD contain only a limited number of options. All the available options can be found in the folder `$MESA_DIR/star/defaults`. Compile the code with `./mk` and run the executable `./rn`, which will compute the stellar model according to the options set in the inlist files.

   (c) MESA has the ability of plotting important quantities while the model is running. These plots are shown according to the options set in the `pgstar` inlist. When the model finishes running, detailed outputs can be found in the `LOGS` for the history of the simulation (`history.data`) and a temporal snapshot for each profile (`profile**.data`). Studying the inlists, outputs and live plots, can you describe which type of star did you just simulated?

[1]Suggestion: If you are working in the laboratory computer create a folder with your name in the documents folder and do all your work there.

2. In order to obtain the eigen modes of oscillation for the Sun we will need to use the stellar evolution code to model the Sun.

   (a) Use the provided `solar` WD to create a solar model. Copy the `solar` WD to your work environment and compile it.

   (b) Edit the `inlist_solar` file, fill in the missing input values accordingle and run the executable `rn` to obtain a Solar model.

   (c) Study the output files (a final profile should be found in the WD root directory) and compare the values obtained for the Effective temperature and Radius at present day with observations.

3. So far, you used one of the available initial heavy element abundances. However, this input parameter can be chosen by the user, and will have considerable impact in the final solar model.

   (a) Repeat step **2.** this time choosing a different heavy abundance mixture by editing the `&control` inlist in the `inlist_project` file:

   ```
   ...
   kappa_file_prefix = 'a09'
   kappa_lowT_prefix = 'lowT_fa05_a09p'
   ...
   ```

   (b) Using the output files Plot the Sound speed profile for each mixture and compare with the observed profile.

   Suggestion: Plot the difference between the obtained SSP and the observed one

   $$\frac{\delta c}{c} = \frac{c_{\rm obs} - c_{\rm model}}{c_{\rm model}}. \tag{1}$$

   for each model. The observed SSP can be found in Basu et al. (2009), ArXiv:0905.0651.

   (c) From the analysis of the curves obtained in the last point, choose one of the input heavy abundances, and justify your choice.

4. Obtain the oscillation frequencies for the chosen solar model.

   (a) The package *GYRE* uses the output of mesa (`solar.mesa`) and computes the eigenfrequencies for the corresponding model. Run it using the command inside the WD

   ```
   $GYRE_DIR/bin/gyre gyre.in
   ```

   (b) A summary of the the computed frequencies can be found in the file `freqs_summary`. Plot the frequencies in a frequency vs. radial order l plane.

Annex I: Useful commands for Terminal

- Change current directory:
  `> cd [desired directory]`
  If you want to refer to the previous directory you can use:
  `>cd ..`

- Sisplay content of directory:
  `>ls [desired directory]`
  Typing ls without arguments will display the content on the actual directory.

- Move or rename file/directory:
  `>mv [current name/directory] [new name/directory]`

- Secure Shell Host (SSH) connection:
  `>ssh [user]@[address]`
  Within the SSH connection you can use all the usual commands to navigate between directories, execute and read files (depending on the permissions), etc..

- Secure File Transfer Protocol (SFTP):
  `>sftp [user]@[address]`
  Within the SFTP environment, you can use the basic commands to navigate between directories in the connected server. To change directory and display directory contents in the local machine, use
  `>lcd [desired local directory]`
  and
  `>lls [desired local directory]`

  - Download files within SFTP:
    `>get [desired file in server host]`
  - Upload files within SFTP:
    `>put [desired file in local machine]`

- Display text file content:
  `>more [file]`
  To continue viewing file press Enter. To stop, press ctrl+c.

- Editing text files:
  `>nano [file]`
  Nano is a text processor built in the terminal, and its controls are explained in the terminal window.

- Read the manual for a desired command:
  `>man [command]`
  This will command will display the flags and options available for the desired command.

Annex II: Useful commands MatLab

- Import data:

  importdata(`filename`,`delimiter`,`Header lines`)

  - `filename` has to be inside quotes, and is the name of the data file (if it is not on the working directory, you need to specify it's directory).
  - `delimiter` is a string and tells MatLab the delimiter between the data values.
  - `Header lines` is an integer, and tells MatLab the number of header lines in file `filename`.

- Interpolate data:

  `vq` = interp1(`x`,`v`,`xq`,`method`)

  - `vq` is the interpolated data from (`x`, `v`) at the query points `xq`.
  - `method` is the method of interpolation. It can be: `linear`, `cubic`, `spline`, etc.

- Plot data:

  plot(`x1`,`y1`,`LineSpec1`,`x2`,`y2`,`LineSpec2`,...)

  - Plot the curves (`x1`,`y1`), (`x2`,`y2`) etc.
  - `LineSpec` sets the line style, marker symbol, and colour. For example (...,`"--"`,`"r"`,...) will plot a red dashed line.