MESA Summer School 2023 Lab Bugnet

Lisa Bugnet, Cole Johnston, Arthur Le Saux, Joey Mombarg

June 2023

Exercise instructions

0.1 Minilab 1 (Arthur)

This first minilab focuses on the coupling between p and g modes, i.e. mixed modes, in a red giant star models. To illustrate this coupling, a useful diagnostic is to look at the mode inertia, which is defined as (look for ref)

$$I = \frac{1}{M} \int_0^M \left[\xi_r^2(r) + \ell\,(\ell+1)\,\xi_{\mathrm{h(r)}}^2 \right] \mathrm{d}m, \tag{1}$$

with M the mass of the star, ξ_r and ξ_{h} the radial and horizontal displacements associated with the mode, ℓ the angular degree from the spherical harmonics and $\mathrm{d}m = 4\pi\rho r^2 \mathrm{d}r$ the mass enclose in the sphere between r and $r+\mathrm{d}r$. For mixed modes, the mode inertia presents a typical oscillating pattern as illustrated in Fig. 1, which present the inertia I of mixed modes as a function of frequency. Each cross corresponds to an eigenmode computed using GYRE. The modes with low inertia are mostly confined in the envelope and thus dominated by the p mode component, whereas the modes with high inertia are dominated by the g mode part. The high density of the modes in the crest of the oscillations are a consequence of the asymptotic period spacing of g modes $\Delta\Pi$ (already introduced on Tuesday).

0.1.1 Evolve model to RGB

For this minilab, the aim is to run a model of 1.4 $M_\odot$ star from a pre-computed ZAMS model up to the Red Giant Branch (RGB) and then to modify the `run_star_extras` file to run GYRE on the fly during a MESA run. First, download the `minilab_1` work directory. The `inlist_project` file from this working directory has already been edited to run from ZAMS and to stop when the centre of the star is depleted in hydrogen. As usual, start by changing the current working directory and compile the code, with

```
cd minilab_1
./mk
```

This step should create the *star* executable file. You can run the model, i.e. make the star evolve, using the command

```
./rn
```

A PGstar plot window displaying information about the current and past state of the star should appear. You can stop the run using the command `Ctrl+C`. Have a look at the `inlist_project` to see what settings are used for this run. Ask one of the TAs if there is anything you don't understand.

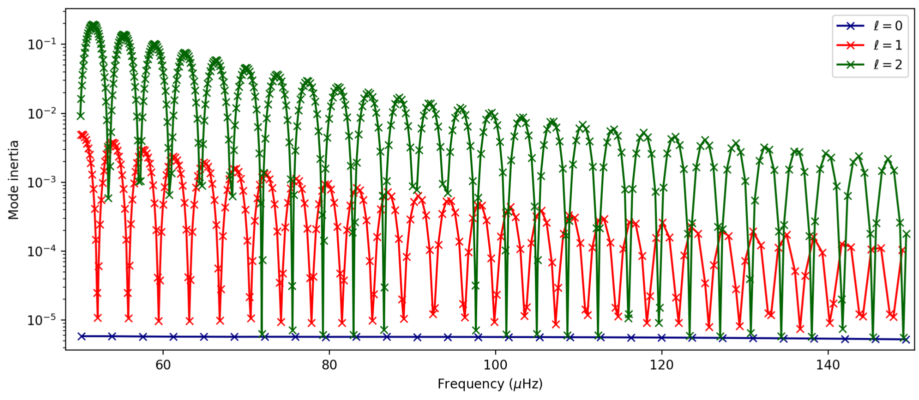

Figure 1: Mode inertia for a RG model, characteristic of mixed modes. Computed with GYRE outside MESA.

0.1.2 Running GYRE on the fly

As mentioned, we want to observe the variations of the mode inertia as a function of frequency, as the star evolve to identify mixed modes. To do so, we want to run GYRE at each time step during the MESA run. This is done by editing the `run_star_extras` file. Open the `run_star_extras` file that is located in the `src/` directory. To initialize GYRE, add the following line in the `run_star_extras`

```
    ! Initialize GYRE

     call gyre_init('gyre_mix.in')

    ! Set constants

    call gyre_set_constant('G_GRAVITY', standard_cgrav)
    call gyre_set_constant('C_LIGHT', clight)
    call gyre_set_constant('A_RADIATION', crad)

    call gyre_set_constant('M_SUN', Msun)
    call gyre_set_constant('R_SUN', Rsun)
    call gyre_set_constant('L_SUN', Lsun)

    call gyre_set_constant('GYRE_DIR', TRIM(mesa_dir)//'/gyre/gyre')
```

The first function `gyre_init` initialises GYRE and calls the inlist file `gyre_mix.in`. This file is used to set up the parameters required to run GYRE and is already present in the working directory.

Next, in order to run GYRE we have added a subroutine `run_gyre` at the end of the `run_star_extras` file.

```
subroutine run_gyre (id, ierr)

    integer, intent(in)  :: id
```

```fortran
integer, intent(out) :: ierr

real(dp), allocatable :: global_data(:)
real(dp), allocatable :: point_data(:,:)
integer               :: ipar(0)
real(dp)              :: rpar(0)

! Pass model data to GYRE

call star_get_pulse_data(id, 'GYRE', .FALSE., .TRUE., .FALSE., &
    global_data, point_data, ierr)
if (ierr /= 0) then
    print *,'Failed when calling star_get_pulse_data'
    return
end if

call gyre_set_model(global_data, point_data, 101)

! Run GYRE to get modes

call gyre_get_modes(1, process_mode, ipar, rpar)

 gyre_has_run = .true.

 contains

subroutine process_mode (md, ipar, rpar, retcode)

    type(mode_t), intent(in) :: md
    integer, intent(inout)   :: ipar(:)
    real(dp), intent(inout)  :: rpar(:)
    integer, intent(out)     :: retcode
    integer :: k

    type (star_info), pointer :: s
    ierr = 0
    call star_ptr(id, s, ierr)
    if (ierr /= 0) return

    ! Print out degree, radial order, mode inertia, and frequency
    print *, 'Found mode: index, l, m, n_p, n_g, zeta, nu = ', &
        md%id-nmax_prev, md%l, md%m, md%n_p, md%n_g, md%n_pg,
        REAL(md%E_norm()),REAL(md%freq('UHZ'))


    frequencies(md%l+1, md%id-nmax_prev) = REAL(md%freq('UHZ'))
    inertias(md%l+1, md%id-nmax_prev) = REAL(md%E_norm())
    nmax = md%id
```

```
        retcode = 0
    end subroutine process_mode


end subroutine run_gyre
```

This subroutine runs GYRE on a given MESA model identified with the variable `id`. First, the function `star_get_pulse_data` extract from the MESA model the data required for pulsation analysis. These data are separated in two arrays: `global_data` and `point_data`. Next, the function `gyre_set_model` sends these data to GYRE. Then, with the function `gyre_get_modes`, GYRE actually computes the eigenmodes of the stellar model for angular degree $\ell = 1$. In this function the first integer indicates the angular degree to compute, it can be modified to get other modes. This function takes as an argument `process_mode`, which is the last subroutine we have defined. It means that when executing the function `gyre_get_modes`, MESA calls and execute `process_mode`. Thanks to this function, we can decide what GYRE outputs are. Here, we are interested in the frequencies and the inertia of the modes, and we store them in the global arrays `frequencies(:,:)` and `inertias(:,:)`.

Now that we have set up GYRE, it is ready to run during a MESA run, the last thing to do is to set:

```
    x_logical_ctrl(1) = .true.
```

in the `inlist_project` file. Then, start running MESA with the usual command `./rn`. At some point during the run, the terminal should print something like

```
Found mode: index, l, m, n_p, n_g, zeta, nu = 130 1 0 3 82 -79 2.285E-02 125.57
```

The last variable *nu* is the frequency of the corresponding mode. You can edit the `gyre_mix.in` to change the range of frequencies of the modes computed by GYRE. To do so change the parameters

```
    freq_min = 50
    freq_max = 150
```

0.1.3 Mode inertia

The last step for this minilab is to plot the mode inertia to see what it looks like. For that, we need to edit the `inlist_pgstar` file. This file controls what is plotted in the pgstar window during a MESA run. Take some time to have a look at it. You can notice that there are parameters to control what is plotted but also the size and location of the plots. In order to get the mode inertia plotted, the next lines of code should be added at the end of the file

```
  ! Add mode inertia panel

  Grid1_plot_name(6) = 'Profile_Panels1' !
  Grid1_plot_row(6) = 5
  Grid1_plot_rowspan(6) = 4
  Grid1_plot_col(6) = 5
  Grid1_plot_colspan(6) = 6

  Profile_Panels1_num_panels = 1
  Profile_Panels1_title = 'Mode inertia l=1'
  Profile_Panels1_xaxis_name = 'freq_l1'
```

```
Profile_Panels1_yaxis_name = 'Enorm_l1'
Profile_Panels1_other_yaxis_name(1) = ''
Profile_Panels1_xmin = 50
Profile_Panels1_xmax = 150

Grid1_plot_pad_left(6) = 0.05
Grid1_plot_pad_right(6) = 0.05
Grid1_plot_pad_top(6) = 0.04
Grid1_plot_pad_bot(6) = 0.07
Grid1_txt_scale_factor(6) = 0.5
```

The two parameters `Profile_Panels1_xaxis_name` and `Profile_Panels1_xaxis_name` define what variable are assigned to the x and y axis respectively. You can modify the displayed range of frequency by editing the `Profile_Panels1_xmin` and `Profile_Panels1_xmax` parameters. Verify that you get the oscillations pattern presented in Fig. 1 for the dipolar mode $\ell = 1$.

0.1.4 Bonus exercise

Try to plot the mode inertias for the modes of spherical degree $\ell = 0$ or 2. To do so, you will need to edit the following files: `gyre_mix.in`, `run_star_extras` and `inlist_pgstar`.

0.2 Minilab 2 (Cole)

In this Minilab, we want to investigate how rotation modifies the structure, and hence, the asteroseismic signature of a typical red giant. We will follow two cases: 1) Where we impose a constant viscosity to approximate rigid rotation, and 2) where we impose a rotation rate at the zero-age main-sequence (ZAMS) of 20% the critical rotation rate. Both of these will require modifications to the standard inlist that we will follow below.

0.2.1 Approximating uniform rotation with high viscosity

We need to make some small additions to the inlist. Specifically, we need to tell MESA how to deal with rotation. Since we are using a large viscosity to approximate the imapct of rotation, we want to set the following parameters.

First, we need to set some flags in the star_job section of the inlist so MESA knows that we want to activate rotation.

```
! Rotation

new_rotation_flag = .true.
change_rotation_flag = .false.
change_initial_rotation_flag = .true.
```

We tell MESA that we want to set a new rotation rate, but we only want to set a new rotation rate at the beginning of the run. Therefore, we set the change_initial_rotation$_f$lag to .true..

Instead of doing complicated maths to figure out what the appropriate rotation rate is in radians per second, we will utilise the functionality of MESA to set the rotation rate as a fraction of the critical rotation rate.

```
near_zams_relax_omega_div_omega_crit = .true.

set_omega_div_omega_crit = .false.
set_initial_omega_div_omega_crit = .true.
new_omega_div_omega_crit = 0.1d0

num_steps_to_relax_rotation = 50

change_D_omega_flag = .true.
new_D_omega_flag = .true.
```

Now that we've set the options in the star_job section, we need to set the options in the controls section. In this section, we only want to approximate the influence of rotation assuming a high diffusive viscosity which will mimic rigid body rotation. This can be achieved by setting the following options:

```
    set_uniform_am_nu_non_rot = .true.
    uniform_am_nu_non_rot = 1d20
```

For the sake of time, we want to make sure that we don't evolve our star all the way to core Helium exhaustion. So we're going to modify the run_star_extras.f90 file to terminate when the model reaches $\nu_{\rm max} = 185\,\mu$Hz. To do this, we need to impose a new termination code in the extras_finish_step function.

```
  if (s% nu_max < 180.) extras_finish_step = terminate
```

In the next step, we'll be passing the stellar profiles to GYRE. To do that, we need MESA to output profiles at each time-step. What we do not want, however, is to clog up our computers with tons of profiles, so, we'll set a condition in the `run_star_extras` to only write profiles when the model is on the red giant branch with pulsations near $\sim\ 200\,\mu$Hz. We can include this condition in the same place as our last condition in the extras_finish_step function.

```
  if (s% nu_max .lt. 250.) s% write_profiles_flag = .true.
```

In the GYRE inlist, we set

```
    Omega_rot_source = 'MODEL'
```

Using this option, GYRE will use the rotation profile of the MESA model to account for the effect of rotation on the stellar pulsations. Next, tell GYRE which MESA model to use as input

```
    file = 'xxx.data.GYRE'
```

and give a name for the output (summary) file

```
    summary_file = 'xxx.data'
```

0.2.2 Angular momentum transport through (magneto)hydrodynamical processes

Now, we want to take a more physical approach and compute the viscosity from the six (magneto)hydrodynamical processes implemented in MESA that can induce turbulence (and thus transport angular momentum). First, we now disable using a uniform viscosity in the inlist

```
set_uniform_am_nu_non_rot = .false.
```

In MESA, each process can be turned on and off separately. To enable all of them without any additional scaling, set all diffusion coefficients equal to 1,

```
D_DSI_factor = 1
D_SH_factor  = 1
D_SSI_factor = 1
D_ES_factor  = 1
D_GSF_factor = 1
D_ST_factor  = 1
```

Run GYRE again at the same age, and compare the pulsations. Could asteroseismology possibly distinguish between these two cases?

0.3 Maxilab (Joey)

In this Maxilab, we are going to infer the internal magnetic field of the red giant (RG) KIC11515377, observed with the NASA *Kepler* mission. We follow the methodology of Li et al. (2022, Nature). The squared radial magnetic field averaged in the horizontal direction is inferred by,

$$\langle B_r^2 \rangle = \frac{\mu_0 \delta\omega_g (2\pi\nu_{\max})^3}{\mathcal{I}}, \tag{2}$$

where $\mu_0 = 4\pi \cdot 10^{-6}\,\mathrm{kG\,cm\,A^{-1}}$ is the magnetic permeability in vacuum, $\delta\omega_g$ is the observed frequency shift of the g modes, and $\nu_{\max}$ is the large frequency separation. The factor $\mathcal{I}$ in the denominator is defined as,

$$\mathcal{I} = \frac{\int \left(\frac{N}{r}\right)^3 \frac{dr}{\rho}}{\int \frac{N}{r} dr}. \tag{3}$$

The denominator in the integral relates to the asymptotic period spacing of modes with spherical degree $\ell = 1$ as follows,

$$\Delta\Pi_{\ell=1} = \frac{2\pi^2}{\sqrt{2}} \left(\int \frac{N}{r} dr \right)^{-1}. \tag{4}$$

We are going to compute the quantity $\mathcal{I}$ of a MESA model in the `run_star_extras`.

- Exercise 0: As a first step, copy the work directory over and verify it runs.

At each step of the evolution we want to compute $\langle B_r^2 \rangle^{1/2}$ and store it in the output of the history file.

- Exercise 1: Prepare in the `run_star_extras` three additional history columns named 'I', 'Br_mean', and 'Delta_Pi1'. Set the correct number of additional columns in the `how_many_extra_history_columns` function, and add the following to `data_for_extra_history_columns` for each additional column,

```
    names(1) = '...'
    vals(1) = ...
```

You can set the values to 0 for now. Do a `./clean` and `./mk` and check this works.

- Exercise 2: The first step is to compute the two integrals in Eq (3). For the Brunt-Väisälä frequency, we need to first ensure it is zero in convective regions and so we compute a new array with all elements ≥ 0. A new array of a variable length is defined as follows,

```
double precision, allocatable :: brunt_N(:)

allocate(brunt_N(s% nz))
```

This defines an array with the same length as the number of cells at each time step. Compute N from the values of N^2 defined in MESA, but set negative values to zero. Afterwards, you can deallocate the array to free up memory.

```
deallocate(brunt_N)
```

In MESA, there are quantities that are defined at the mass centre of the cell, and there are quantities that are defined at the edge of the cell. Think about this when you compute the integrals. If your model has a high enough spatial resolution, you can assume,

$$\int x \,\mathrm{d}x \approx \sum_i x_i \,\Delta x_i, \tag{5}$$

where the index i runs over the cells.
Hint: In `star_info`, `s% r` is defined at the cell edge, while `s% rmid` is defined at the centre.

Once you have computed $\mathcal{I}$, write this value out to the first extra column in history.

- Exercise 3: Next, we want to pass on the value of $\delta\omega_g$ to the `run_star_extras`. In your inlist, you can set

```
    x_ctrl(1) = ...
```

to a value that you can then access in the `run_star_extras` through,

```
    s% x_ctrl(1)
```

Add a control in your inlist to do this. The observed value for KIC11515377 is $\delta\omega_g/(2\pi) = 126\,\mathrm{nHz}$. The value of ν_{max} you can get from `star_info`. Finally, write $\langle B_r^2 \rangle^{1/2}$ and $\Delta\Pi_1$ also to your history file. Recompile and verify that on the RGB you find an average magnetic field of the order of 100 kG.

- Exercise 4: Finally, we want to stop the evolution when the model has roughly reached the observed values of $\nu_{\mathrm{max,obs}} = 191.6 \pm 1\,\mu\mathrm{Hz}$ and $\Delta\Pi_{1,\mathrm{obs}} = 83.16 \pm 1\,\mathrm{s}$. Add two additional controls to your inlist to pass these two values on to the `run_star_extras` and define

$$\chi^2 = (\nu_{\mathrm{max}} - \nu_{\mathrm{max,obs}})^2 + (\Delta\Pi_1 - \Delta\Pi_{1,\mathrm{obs}})^2. \tag{6}$$

Change the inlist to start the evolution from the zero-age main sequence instead of loading in a precomputed RGB model. (Be sure to properly set the initial composition!) Once on the RGB, after each time step, check whether the χ^2 is smaller or bigger than the previous value. If it is bigger, terminate. Add to your PGstar inlist the target values, so that you can see how close your models gets to the observations. Pick a value for the initial mass from the spreadsheet and note down the lowest found χ^2 value and the corresponding value of the internal magnetic field (in kG).

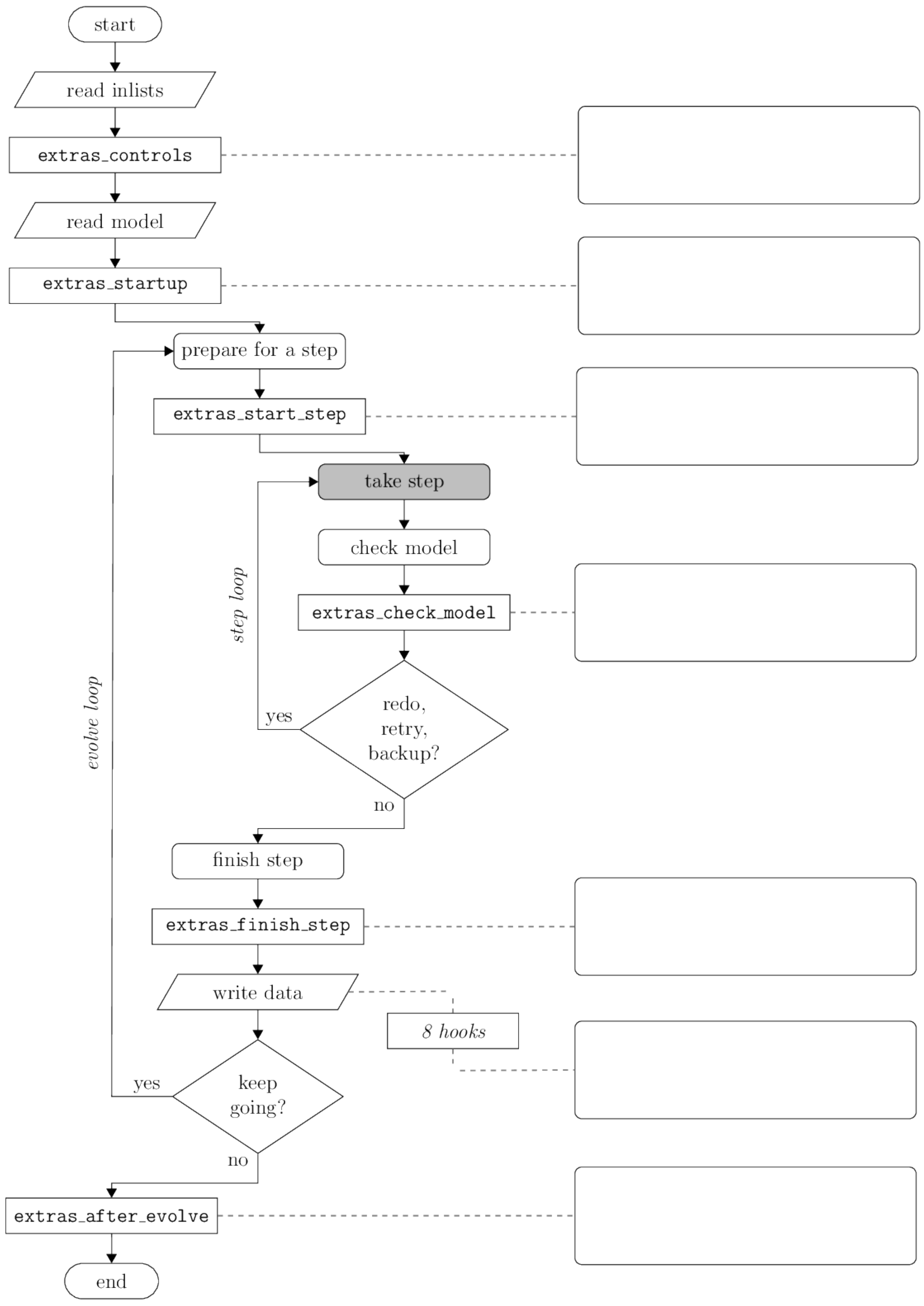

Figure 2: Flowchart of the `run_star_extras`.