Numerical computation of adiabatic stellar pulsation

Masao Takata

Department of Astronomy, School of Science, The University of Tokyo

27 February 2025

Asymptotics in astrophysics iTHEMS workshop

Software

- stellar oscillations
  - **ADIPLS** by J. Christensen-Dalsgaard
    https://phys.au.dk/~jcd/adipack.v0_3b/
  - **GYRE** by R. H. Townsend
    https://gyre.readthedocs.io/

- evolutionary stellar models
  - **MESA** by B. Paxton and collaborators, which includes ADIPLS and GYRE
    https://docs.mesastar.org/
  - standard solar model (**Model S**) by Christensen-Dalsgaard et al. (1996)
    https://phys.au.dk/~jcd/solar_models/

Equations to solve

- fundamental equations
  - linearised mass conservation
  - linearised equation of motion
  - linearised Poisson equation
  - adiabatic relation
- separation of variables
  - temporal dependence: $\exp\left(i\sigma t\right)$
  - angular dependence: $Y_\ell^m\left(\theta,\,\phi\right)$ (spherical harmonics)
  - radial dependence (r): 4th-order system of ordinary differential equations

Radial part variables

displacement vector

$$\boldsymbol{\xi} = \left(\xi_r(r) Y_\ell^m,\ \xi_h(r) \frac{\partial Y_\ell^m}{\partial \theta},\ \xi_h(r) \frac{1}{\sin\theta} \frac{\partial Y_\ell^m}{\partial \phi} \right)$$

dependent variables (Dziembowski variables):

$$y_1 = \frac{\xi_r}{r}, \quad y_2 = \frac{1}{gr}\left(\frac{p'}{\rho} + \Phi' \right) = \frac{\sigma^2}{g}\xi_h, \quad y_3 = \frac{\Phi'}{gr}, \quad y_4 = \frac{1}{g}\frac{\mathrm{d}\Phi'}{\mathrm{d}r}$$

(squared) dimensionless frequency:

$$\omega^2 = \frac{R^3\sigma^2}{GM},$$

equilibrium structure variables:

$$c_1 = \frac{(r/R)^3}{M_r/M}, \quad U = \frac{4\pi r^3 \rho}{M_r}, \quad V_g = -\frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} = \frac{gr}{c^2}, \quad A^* = \frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} - \frac{\mathrm{d}\ln \rho}{\mathrm{d}\ln r}$$

g : gravitational acceleration; p : pressure; ρ : density; Φ : gravitational potential; c : sound speed

M : total mass; R : total radius; G : gravitational constant

Two-point boundary value problem with eigenvalue ω^2

$$r\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}r} = A\boldsymbol{y} \quad \text{with} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}, \quad A = \begin{pmatrix} V_g - 3 & \dfrac{\ell(\ell+1)}{c_1\omega^2} - V_g & V_g & 0 \\ c_1\omega^2 - A^* & A^* - U + 1 & -A^* & 0 \\ 0 & 0 & 1 - U & 1 \\ UA^* & UV_g & \ell(\ell+1) - UV_g & -U \end{pmatrix}$$

boundary conditions

- at the centre ($r = 0$): regularity conditions

$$c_1\omega^2 y_1 - \ell y_2 = 0\,, \quad \ell y_3 - y_4 = 0$$

- at the surface ($r = R$):

$$y_1 - y_2 + y_3 = 0\ (\delta p = 0)\,, \quad U y_1 + (\ell+1)\, y_3 + y_4 = 0\ (\text{matching } \Phi' \text{ in the outside})$$

Methods to solve

- **shooting method**
  1. set a trial value of ω^2
  2. integrate $\boldsymbol{y}$ from $r = 0$ to $r = r_{\rm fit}$ to obtain $\boldsymbol{y}_{\rm a}$
  3. integrate $\boldsymbol{y}$ from $r = R$ to $r = r_{\rm fit}$ to obtain $\boldsymbol{y}_{\rm b}$
  4. check if $\boldsymbol{y}_{\rm a}$ matches with $\boldsymbol{y}_{\rm b}$ or not

- **relaxation method** (or Henyey method)
  1. set a trial value of ω^2 and a trial profile of $\boldsymbol{y}$
  2. determine $\Delta\boldsymbol{y}$ and $\Delta\omega^2$ so that $\boldsymbol{y} + \Delta\boldsymbol{y}$ and $\omega^2 + \Delta\omega^2$ satisfy the finite difference equations
  3. check if the difference $\Delta\boldsymbol{y}$ and $\Delta\omega^2$ are both small enough or not

The problem is often reduced to the root-finding problem of a discriminant $D(\omega)$.

Demonstration using GYRE

Townsend, R.H., Teitler, S.A., 2013, MNRAS, 435, 3406

1. download and install MESA SDK
   `http://user.astro.wisc.edu/~townsend/static.php?ref=mesasdk`
2. download and install GYRE
   `https://gyre.readthedocs.io/`
3. install the PYTHON package, pygyre
   `https://pygyre.readthedocs.io/`
4. edit the parameter file gyre.in
5. run gyre
6. check the results

step 1/6: MESA SDK

http://user.astro.wisc.edu/~townsend/static.php?ref=mesasdk

- available for Linux (Intel/AMD) and MacOS (Intel and ARM)
- follow the installation section of the page

step 2/6: GYRE

https://gyre.readthedocs.io/

- download the source code

```
wget -c https://github.com/rhdtownsend/gyre/releases/download/v7.2.1/gyre-7.2.1.tar.gz
```

- unpack and install

```
tar xfz gyre-7.2.1.tar.gz -C /some/where
export GYRE_DIR=/some/where/gyre-7.2.1
make -j -C $GYRE_DIR install
```

step 3/6: pygyre

https://pygyre.readthedocs.io/

- install PYTHON on your platform (if you have not done yet)
- install pygyre

```
pip install pygyre
```

step 4/6: edit gyre.in (1)

- make a working directory

```
mkdir /some/where/work
```

- copy the template (for a solar model)

```
cp -r $GYRE_DIR/test/ad/fgong/solar /some/where/work
```

- edit gyre.in

```
cd /some/where/work/solar
vi gyre.in
```

step 4/6: edit gyre.in (2)

possible changes

- spherical degree ℓ

```
&mode
    l = 1000
/
```

$\Rightarrow$

```
&mode
    l = 1
/
```

- frequency range

```
&scan
  grid_type = 'LINEAR'
  freq_min = 3000
  freq_max = 5000
  freq_min_units = 'UHZ'
  freq_max_units = 'UHZ'
  n_freq = 10
/
```

$\Rightarrow$

```
&scan
  grid_type = 'LINEAR'
  freq_min = 1000
  freq_max = 5000
  freq_min_units = 'UHZ'
  freq_max_units = 'UHZ'
  n_freq = 100
/
```

step 4/6: edit gyre.in (3)

possible changes

- outputs (to include eigenfunctions)

```
&ad_output
  summary_file = 'summary.h5'
  summary_item_list = 'l,n_pg,n_p,n_g,freq'
  freq_units = 'UHZ'
/
```

⇒

```
&ad_output
  summary_file = 'summary.h5'
  summary_item_list = 'l,n_pg,n_p,n_g,freq'
  detail_template = 'detail.l%l.n%n.h5'
  detail_item_list = 'l,n_pg,omega,x,
          xi_r,xi_h,eul_P,eul_rho,eul_Phi,
          lag_P,lag_rho,
          c_1,U,V_2,As,Gamma_1,rho,P'
  freq_units = 'UHZ'
/
```

step 5/6: run GYRE

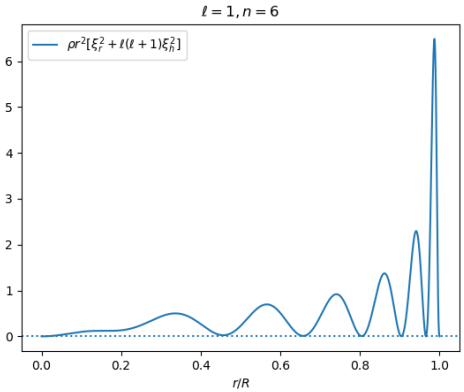
```
cd /some/where/work/solar
$GYRE_DIR/bin/gyre gyre.in
```

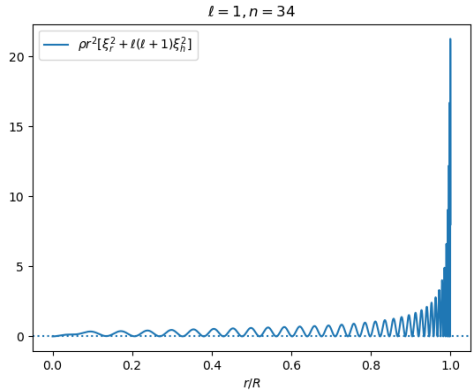
output files:

- summary file
  `summary.h5`
- detail files
  `detail.l1.n+6.h5, detail.l1.n+7.h5, ..., detail.l1.n+34.h5`

step 6/6: check the results

- use `pygyre` to manipulate the results
- try `plot_detail.py`





That’s all, thank you!

Appendix

More about the shooting method

- start integration from $r = r_\epsilon (> 0)$ and $r = R - r_\epsilon (< R)$ to avoid singularities
  - two linearly-independent solutions near the centre

$$\boldsymbol{y}_1 \approx r^{\ell-2} \begin{pmatrix} \ell \\ (c_1)_{\text{centre}}\, \omega^2 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{y}_2 \approx r^{\ell-2} \begin{pmatrix} 0 \\ 0 \\ 1 \\ \ell \end{pmatrix}$$

  - two linearly-independent solutions near the (zero-boundary) surface

$$\boldsymbol{y}_3 \approx \begin{pmatrix} 1 \\ 1 \\ 0 \\ -(U)_{\text{surface}} \end{pmatrix}, \quad \boldsymbol{y}_4 \approx \begin{pmatrix} 0 \\ 1 \\ 1 \\ -\ell - 1 \end{pmatrix}$$

- match condition at the fitting point $r = r_{\text{fit}}$

$$D\left(\omega^2\right) = \det\left(\boldsymbol{y}_{\text{a},1}\left(r_{\text{fit}}\right) \quad \boldsymbol{y}_{\text{a},2}\left(r_{\text{fit}}\right) \quad \boldsymbol{y}_{\text{b},3}\left(r_{\text{fit}}\right) \quad \boldsymbol{y}_{\text{b},4}\left(r_{\text{fit}}\right)\right) = 0$$

zeros of $D\left(\omega^2\right)$ provide the (squared) eigenfrequencies ω^2

Comments on the frequency scanning

need to know the properties of the solutions in advance to adjust the parameters for the frequency scanning

- estimate the dynamical time scale: $t_{\rm dyn} \sim \left(\frac{GM}{R^3}\right)^{-\frac{1}{2}}$
- decide which type of modes to compute (p or g)
- adjust the parameters as follows:

p modes	g modes
`freq_min`, `freq_max` $\gtrsim t_{\rm dyn}^{-1}$	`freq_min`, `freq_max` $\lesssim t_{\rm dyn}^{-1}$
`d_type = 'LINEAR'`	`d_type = 'INVERSE'`
`n_freq` $\gg \frac{\texttt{(frequency range)}}{\Delta\nu}$	`n_freq` $\gg \frac{\texttt{(period range)}}{\Delta\Pi}$