GYRE

Release master

Jul 31, 2020

User Guide

GYRE is a *stellar oscillation code*. Given an input stellar model, GYRE calculates the eigenfrequencies and eigenfunctions for the normal oscillation modes of the model. These data can be put to a variety of uses; the most common is to compare them against observed oscillation frequencies of a star, allowing constraints on the star's fundamental parameters (mass, radius, etc.) to be established — the discipline of *asteroseismology*.

CHAPTER 1

About this Manual

This manual is divided into two main parts: the *User Guide*, which covers basic installation and usage of GYRE, and the *Reference Guide*, which documents all of GYRE's options in detail. Supplementary material can be found in the *Appendices*.

1.1 Preliminaries

1.1.1 Intended Audience

This manual is aimed at a broad audience — whether you're a GYRE novice or a seasoned veteran, it provides you with the information you'll need to get the most out of GYRE. However, it does presume some experience with Unix command-line environments, and likewise some basic familiarity with the subject of stellar oscillations. If you need the former, then the Internet is your oyster; and for the latter, we recommend the following online resources:

- Jørgen Christensen-Dalsgaard's Lecture Notes on Stellar Oscillations;
- Gerald Handler's Asteroseismology article.

1.1.2 Obtaining GYRE

The source code for GYRE is hosted in the rhdtownsend/gyre git repository on GitHub. GYRE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3.

1.1.3 Citing GYRE

If you use GYRE in your research, please cite the relevant 'instrument' papers:

- Townsend & Teitler (2013) describes the basic operation of the code;
- Townsend et al. (2018) outlines improvements to allow non-adiabatic calculations.

- Goldstein & Townsend (2020, in press) describes the contour method for non-adiabatic calculations.

If you find yourself using GYRE on a regular basis, you might consider contributing to the project to ensure its long-term success. Options include

- Contribute code to the project (e.g., via GitHub pull requests), to extend GYRE's capabilities
- Contribute documentation and tutorials to the project, to make GYRE more user-friendly
- Invite the GYRE team to be co-authors on relevant papers
- Invite the GYRE team to be co-investigators on relevant grant applications

1.1.4 Related Links

- The GYRE discussion forums, the place to post feature requests and bug reports (don't send emails!).
- The MESA Software Development Kit (SDK), which provides the compilers and supporting libraries needed to build GYRE.
- The MESA Stellar Evolution Code, which can generate stellar models readable by GYRE.

1.1.5 Acknowledgements

GYRE has been developed with financial support from the following grants:

- NSF awards AST-0908688, AST-0904607, ACI-1339606, ACI-1663696, and AST-1716436;
- NASA awards NNX14AB55G, NNX16AB97G, and 80NSSC20K0515.

GYRE has also benefitted greatly from contributions (code, bug reports, feature requests) from the academic community.

1.2 Quick Start

To get started with GYRE, follow these five steps:

1. Install the MESA Software Development Kit
2. Download the GYRE source code
3. Unpack the source code using the **`tar`** utility
4. Set the `GYRE_DIR` environment variable to point to the newly created source directory
5. Compile GYRE using the command **`make -C $GYRE_DIR`**

For a more in-depth installation guide that covers alternative use-cases, refer to the *Installation* chapter. If the code doesn't compile properly, consult the *Troubleshooting* chapter. Otherwise, proceed to the next chapter where you'll put together your first GYRE calculation.

1.3 Example Walkthrough

This chapter provides a walkthrough of a example GYRE project, to illustrate the typical steps involved. For this example, we'll be focusing on finding eigenfrequencies and eigenfunctions of quadrupole ($\ell = 2$) gravity modes for a slowly pulsating B (SPB) stellar model.

1.3.1 Making a Place to Work

When starting a new project, it's a good idea to create a dedicated working directory to contain the various input and output files that GYRE operates on. These commands will make a new directory beneath your home directory with the name file:*work*, and then change into it:

1.3.2 Grabbing a Stellar Model

The next step is to grab a stellar model for GYRE to work with. There are a number of models provided beneath the `$GYRE_DIR/models` directory; the following commands will copy a MESA model for a 5 $M_\odot$ SPB star into your working directory:

1.3.3 Creating a Namelist File

Now comes the fun part: creating an input file containing the various parameters which control a GYRE run. Using a text editor, create the file `gyre.in` in your working directory with the following contents cut-and-pasted in:

```
&constants
/

&model
        model_type = 'EVOL'  ! Obtain stellar structure from an evolutionary model
        file = 'spb.mesa'    ! File name of the evolutionary model
        file_format = 'MESA' ! File format of the evolutionary model
/

&mode
        l = 2                ! Harmonic degree
/

&osc
        outer_bound = 'VACUUM' ! Use a zero-pressure outer mechanical boundary␣
↪condition
/

&num
        diff_scheme = 'COLLOC_GL4' ! 4th-order collocation scheme for difference␣
↪equations
/

&scan
        grid_type = 'INVERSE' ! Scan for modes using a uniform-in-period grid; best␣
↪for g modes
        freq_min = 0.5        ! Minimum frequency to scan from
        freq_max = 1.0        ! Maximum frequency to scan to
        n_freq = 250          ! Number of frequency points in scan
/

&grid
        alpha_osc = 10  ! Ensure at least 10 points per wavelength in propagation␣
↪regions
        alpha_exp = 2   ! Ensure at least 2 points per scale length in evanescent␣
↪regions
        n_inner = 5     ! Ensure at least 5 points between center and inner turning␣
↪point
```

(continues on next page)

(continued from previous page)

```
/


&ad_output
        summary_file = 'summary.txt'                                ! File name for␣
↪summary file
        summary_file_format = 'TXT'                                 ! Format of summary␣
↪file
        summary_item_list = 'M_star,R_star,l,n_pg,omega,E_norm' ! Items to appear in␣
↪summary file
        mode_template = 'mode.%J.txt'                                  ! File-name␣
↪template for mode files
        mode_file_format = 'TXT'                                    ! Format of mode␣
↪files
        mode_item_list = 'l,n_pg,omega,x,xi_r,xi_h'                    ! Items to␣
↪appear in mode files
/

&nad_output

/
```

This file is an example of a Fortran 'namelist' file, containing multiple namelist groups. Each group begins with the line `&name` (where `name` is the name of the group); a list of name-value pairs follows, and the group ends with a slash `/`. Detailed information on the namelist groups expected in GYRE's input files can be found in the *Namelist Input Files* chapter; for now, let's just focus on some of the more-important aspects of the file above:

- The `&constants` namelist group is used to override constants such as the gravitational constant; here it's empty, indicating that default values should be used
- The `&model` namelist group instructs GYRE to read an evolutionary model, in MESA format, from the file `spb.mesa`
- The `&mode` namelist group instructs GYRE to consider quadrupole ($\ell = 2$) modes
- The `&osc` namelist group instructs GYRE to apply a zero-pressure outer mechanical boundary condition in the oscillation equations
- The `&scan` namelist group instructs GYRE to scan a region of dimensionless angular frequency space typically occupied by gravity modes
- The `&grid` namelist group instructs GYRE to perform calculations on a refinement of the model grid (see the *Working with Grids* chapter for details on how this works)
- The `&ad_output` namelist group instructs GYRE to write out summary data to the file `summary.txt`, and individual mode data to files having the prefix `mode.`
- The `&nad_output` namelist group is empty, telling GYRE not to write out any non-adiabatic data (see non-adiabatic-calculations for more info)

1.3.4 Running GYRE

With the hard work done, it's now trivial to run GYRE:

As the code runs (on multiple cores, if you have a multi-core machine; see faq-multicore for more details), it will print lots of data to the screen. Let's break down this output, chunk by chunk.

First, GYRE prints out its version number, tells us (in OpenMP threads) how many cores it is running on, and indicates which file it is reading parameters from (here, file:*gyre.in*):

```
gyre [5.2]
----------

OpenMP Threads   : 4
Input filename   : gyre.in
```

Next, GYRE loads the stellar model from the file `spb.mesa`. This model comprises 1814 points and extends from the surface all the way to the center (which is why GYRE decides not to add a central point).

```
Model Init
----------

Reading from MESA file
   File name spb.mesa
   File version 1.00
   Read 1814 points
   No need to add central point
```

GYRE then prepares to searching for modes with harmonic degree $\ell = 2$ and azimuthal order $m = 0$ (not specified in `gyre.in`, but assumed by default), by building a frequency grid and a spatial (x) grid:

```
Mode Search
-----------

Mode parameters
   l : 2
   m : 0

Building frequency scan
   added scan interval :  0.5000E+00 ->  0.1000E+01 (250 points, INVERSE)

Building x grid
   Found inner turning points, x range 0.1041 -> 0.1048
   Adding 0 inner point(s)
   Adding 21 global point(s) in iteration 1
   Adding 0 global point(s) in iteration 2
   Final grid has 1 segment(s) and 1835 point(s):
      Segment 1 : x range 0.0000 -> 1.0000 (1 -> 1835)
```

(The concepts of spatial and frequency grids are explored in greater detail in the *GYRE Fundamentals* chapter). Next, GYRE attempts to bracket roots of the discriminant function (again, see *GYRE Fundamentals*) by searching for changes in its sign:

```
Starting search (adiabatic)

Root bracketing
  Time elapsed :     2.122 s
```

Finally, for each sign change found GYRE uses a root solver to converge to the eigenfrequency. Each row of output here corresponds to a mode that GYRE has successfully found:

```
Root Solving
   l    m    n_pg    n_p    n_g       Re(omega)       Im(omega)        chi n_iter
   2    0     -16      0     16  0.51863442E+00  0.00000000E+00 0.3517E-13      6
   2    0     -15      0     15  0.55636128E+00  0.00000000E+00 0.1928E-12      7
   2    0     -14      0     14  0.59457157E+00  0.00000000E+00 0.4534E-13      6
   2    0     -13      0     13  0.62301181E+00  0.00000000E+00 0.9584E-13      7
```

(continues on next page)

(continued from previous page)

```
  2    0     -12      0     12  0.67563541E+00  0.00000000E+00 0.2665E-12      6
  2    0     -11      0     11  0.74334524E+00  0.00000000E+00 0.3755E-13      8
  2    0     -10      0     10  0.79690725E+00  0.00000000E+00 0.3113E-12      6
  2    0      -9      0      9  0.87153108E+00  0.00000000E+00 0.1487E-12      6
  2    0      -8      0      8  0.99747127E+00  0.00000000E+00 0.1265E-12      6
 Time elapsed :      0.597 s
```

The columns appearing are as follows:

l harmonic degree ℓ

m azimuthal order m

n_pg radial order n (in the Eckart-Osaki-Scuflaire-Takata scheme)

n_p acoustic-wave winding number $n_{\rm p}$

n_g gravity-wave winding number $n_{\rm g}$

Re(omega) real part of dimensionless eigenfrequency ω

Im(omega) imaginary part of dimensionless eigenfrequency ω (zero here because we've performed an adiabatic calculation)

chi convergence parameter

n_iter number of iterations required for convergence

These values are printed to screen primarily to give an idea of GYRE's progress; more-detailed information about the modes found is given in the output files discussed below. Some things to watch out for:

- The convergence parameter `chi`, defined as the ratio of discriminant values before and after the root finding, should small (on the order of 1E-9 to 1E-13). If it is significantly larger than this, the mode may not be properly converged; and if it is significantly smaller than this, there may be numerical issues with the discretization scheme.
- The number of iterations `n_iter` should be moderate; values above 20 or so indicate that GYRE is having problems converging.
- The mode index `n_pg` should be monotonic-increasing. Departures from this behavior can happen for a number of reasons:
  - Missing values can indicate that GYRE has skipped a mode in frequency space; the fix is to use a finer frequency scan.
  - Missing values together with duplicate and/or non-monotonic values can indicate that GYRE isn't resolving the spatial structure of eigenfunctions; the fix is to use a finer spatial grid.
  - Missing values together with duplicate and/or non-monotonic values can *also* incdicate problems with the input stellar model — for instance, incorrect values for the Brunt-Vaisala frequency across density discontinuities; the fix is to stop expecting GYRE to give sensible output when fed crap stellar models!

1.3.5 Interpreting Output Files

Overall properties of all modes found (eigenfrequencies, inertias, etc.) are collected together in the file `summary.txt`. For each mode GYRE also writes a file with the name `mode.`*NNNNN*`.txt`, containing data (eigenfrequency, eigenfunctions, etc.) specific to the mode. Here, *NNNNN* denotes a 5-digit index which increments (starting at `00001`) for each mode found. Note that this index bears no relation to the radial order `n_pg`; it merely serves as a unique label for the modes.

Both the sumamry file and the mode files are text-based (it's possible to write HDF5-format files instead; see the *Output Files* chapter for details). The command

will print out the first 10 lines of the summary file, which should look something like this:

```
                                                                                          ␣
↪                                                                                         ␣
↪                                                                                         ␣
↪
                        1                        2
                   M_star                   R_star
  0.9945999999999999E+034  0.3016908790335515E+012
                        1                        2                        3               ␣
↪                 4                        5
                        1                     n_pg                Re(omega)               ␣
↪         Im(omega)                   E_norm
                        2                      -16  0.5186344189658060E+000  0.
↪0000000000000000E+000  0.1083833699332978E-002
                        2                      -15  0.5563612831705178E+000  0.
↪0000000000000000E+000  0.1378396850031897E-002
                        2                      -14  0.5945715662438736E+000  0.
↪0000000000000000E+000  0.3226917642759521E-002
                        2                      -13  0.6230118072964336E+000  0.
↪0000000000000000E+000  0.3598212959967765E-002
```

The first three lines give column numbers, labels, and values for the scalar data — here, the stellar mass `M_star` and radius `R_star`, expressed in cgs units. The next two lines give column numbers and labels for the per-mode data (`E_norm` is the normalized mode inertia, and the other columns are the same as described above for the screen output); the subsequent lines then give the corresponding values (one line per mode). The mode files have a similar layout, with scalar data followed by array data representing the eigenfunctions (one line per radial grid point).

The choice of which data appear in output files isn't hardwired, but rather determined by the `summary_item_list` and `mode_item_list` parameters of the `&ad_output` and `&nad_output` namelist groups. Changing these parameters allows you to tailor the files to contain exactly the data you need. For a full list of possible items, consult the *Output Files* chapter.

1.4 GYRE Fundamentals

This chapter explains the fundamentals of how GYRE works. Although it aims to be user-friendly, GYRE is nevertheless a complex piece of software; thus, getting it to produce the 'best' results requires some degree of insight into the algorithms it uses to caclulate mode eigenfrequencies and eigenfunctions.

1.4.1 The Stretched String Problem

We'll start our discussion of GYRE by considering the analogous (but much simpler) problem of finding normal-mode eigenfrequencies and eigenfunctions for waves on a stretched string clamped at both ends. Let the string have mass per unit length ρ and tension T; then, the wave equation describing the transverse string displacement $y(x,t)$ at spatial position x and time t is

$$yx2 = -\frac{1}{c^2} yt2,$$

with $c \equiv (T/\rho)^{1/2}$. If the string is clamped at $x = 0$ and $x = L$, then the wave equation together with the boundary conditions

$$y(0,t) = 0 \qquad y(L,t) = 0$$

comprise a two-point boundary value problem (BVP).

1.4.2 Analytic Solution

The *stretched-string BVP* is straithgforward to solve analytically. General solutions of the wave equation take the form of traveling waves,

$$y(x,t) = A\exp[(kx - \sigma t)],$$

where A an arbitrary constant, and the frequency σ and wavenumber k are linked by the dispersion relation

$$\sigma^2 = c^2k^2.$$

The phase velocity of these waves is $\sigma/k = \pm c$.

To satisfy the boundary condition at $x = 0$, we combine traveling-wave solutions with opposite-sign wavenumbers

$$y(x,t) = A\exp[(kx - \sigma t)] - A\exp[(-kx - \sigma t)] = B\sin(kx)\exp(-\sigma t),$$

where $B = 2A$. For the boundary condition at $x = L$ to be satisfied simultaneously,

$$\sin(kL) = 0,$$

and so

$$kL = n\pi$$

where n is a non-zero integer (we exclude $n = 0$ because it leads to the trivial solution $y(x,t) = 0$). . Combining this with the dispersion relation, we find the normal-mode eigenfrequencies of the stretched-string BVP are

$$\sigma = n\frac{\pi c}{L}, \tag{1.1}$$

and the corresponding eigenfunctions are

$$y(x,t) = B\sin\left(\frac{n\pi x}{L}\right)\exp(-\sigma t). \tag{1.2}$$

The index n uniquely labels the modes, and $y(x,t)$ exhibits $n-1$ nodes in the open interval $x = (0, L)$.

1.4.3 Numerical Solution

Now let's see how we might go about solving the *stretched-string BVP* numerically, using an approach very similar to the one implemented in GYRE's for solving the oscillation equations.

Separation

We begin by performing a separation of variables on the wave equation, assuming trial solutions of the form

$$y(x,t) = \tilde{y}(x)\exp(-\sigma t),$$

where $\tilde{y}(x)$ is some function of x alone. Then, the wave equation reduces to an ordinary differential equation (ODE) for $\tilde{y}$,

$$\tilde{y}x2 = \frac{\sigma^2}{c^2}\tilde{y}.$$

Discretization

To solve the ODE, we discretize it to form a set of difference equations. The discretization involves transforming the continuous function $\tilde{y}(x)$ into a finite set of N values $\{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_N\}$, representing the function sampled on the discrete spatial grid $\{x_1, x_2, \ldots, x_N\}$.

For simplicity let's assume the grid is uniform, so that

$$x_{k+1} - x_k = \Delta x \equiv \frac{L}{N-1} \qquad (1 \leq k \leq N-1)$$

Then, the second derivative of $\tilde{y}$ can be approximated (to second order in Δx) as

$$\tilde{y}x2|_{x=x_k} \approx \frac{\tilde{y}_{k+1} - 2\tilde{y}_k + \tilde{y}_{k-1}}{\Delta x^2} \qquad (2 \leq k \leq N-1).$$

This allows us to replace the ODE with $N-2$ difference equations

$$\frac{\tilde{y}_{k+1} - 2\tilde{y}_k + \tilde{y}_{k-1}}{\Delta x^2} = \frac{\sigma^2}{c^2}\tilde{y}_k \qquad (2 \leq k \leq N-1)$$

Taken together with the 2 boundary conditions

$$\tilde{y}_1 = 0 \qquad \tilde{y}_N = 0,$$

we have a linear system of N algebraic equations and N unknowns.

Linear System

To find solutions to the linear system, we first write it in matrix form as

$$= \mathbf{0}, \tag{1.3}$$

where is the vector with components

$$= \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_{N-1} \\ \tilde{y}_N \end{pmatrix}$$

and the 'system matrix' is an $N \times N$ tridiagonal matrix with components

$$= \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & \sigma^2\tau^2 - 2 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \sigma^2\tau^2 - 2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}.$$

Here we've introduced

$$\tau \equiv \frac{L}{N-1}c,$$

representing the sound crossing time of a single cell.

Equation (1.3) is a homogeneous linear system, meaning that it only has non-trivial solutions when the determinant of vanishes. With this in mind, we formulate the characteristic equation for the BVP,

$$(\sigma) = 0$$

Fig. 1.1: Plot of the discriminant function (σ) as a function of the frequency σ, for the stretched-string BVP with $N = 50$. The orange dots highlight where $= 0$. The function has been scaled so that $(0) = 1$. (`Source`)

where $(\sigma) \equiv \det()$ is a discriminant function whose roots are the characteristic frequencies (*eigenfrequencies*) of the stretched-string BVP.

Fig. 1.1 plots the discriminant function for the BVP discretized on a spatial grid of $N = 50$ points. The roots (zeros) of the function are highlighted by the orange markers; they fall very close to the values $\sigma = \pi c/L, 2\pi c/L, \ldots$ predicted by the *Analytic Solution*.

Finding Eigenfrequencies

While Fig. 1.1 is useful for visalizing , it's not the best way to find eigenfrequencies. Instead, we can rely on well-established techniques for isolating and refining roots of monovariate functions.

First, we evaluate a finite set of M values $\{_{1,2}, \ldots, _M\}$, representing the discriminant function sampled on the discrete frequency grid $\{\sigma_1, \sigma_2, \ldots, \sigma_M\}$. Then, we inspect the signs of adjacent values $(_j, _{j+1})$. If these differ, then we know that a root of the discriminant function must lie in the interval (σ_j, σ_{j+1}) — we have *bracketed* a root. Fig. 1.2 demonstrates the process of root bracketing for a frequency grid covering the plotted frequency interval with $M = 32$ uniformly spaced points; it highlights five brackets containing the five roots shown previously in Fig. 1.1.

Fig. 1.2: Plot of the discriminant values $\{\}$ on the discrete frequency grid $\{\sigma\}$, for the stretched-string BVP with $N = 50$ and $M = 32$. The orange halos indicate adjacent points that bracket a root $= 0$. (`Source`)

Once a bracket is established for a given root, it can be narrowed through a process of iterative refinement until the root is converged upon. There are a variety of well-known root-finding algorithms that perform this refinement; the bisection method is conceptually the simplest, but approaches such as Brent's method can be much more efficient. For the brackets plotted in Fig. 1.2, Table 1.1 compares the eigenfrequencies found using Python's `scipy.optimize.brentq()` function, against the analytic values predicted by equation (1.1).

Table 1.1: Numerical and analytic eigenfrequencies, in units of $\pi c/L$, for the stretched-string BVP with $N = 50$. (`Source`)

n	numerical	analytic
1	0.999829	1.000000
2	1.998630	2.000000
3	2.995378	3.000000
4	3.989047	4.000000
5	4.978618	5.000000

Eigenfunction Reconstruction

For each of the eigenfrequencies found, we find the corresponding eigenfunction by solving the linear system (1.3). Because $\det()$ is now zero, this system is guaranteed to have a non-trivial solution. The solution vector resides in the null space of , and we can use standard numerical techniques (e.g., singular value decomposition) to evaluate it. Then, the k'th element of corresponds to the eigenfunction sampled at the k'th spatial grid point:

$$()_k = \tilde{y}_k \equiv \tilde{y}(x_k)$$

Fig. 1.3: Plot of the eigenfunctions $\tilde{y}$ as a function of spatial coordinate x, for the first three modes of the stretched-string BVP with $N = 50$. The discrete points show the numerical functions, and the solid lines the corresponding analytic functions. (`Source`)

Fig. 1.3 plots the eigenfunctions found in this way for the first three modes ($n = 1, \ldots, 3$) given in Table 1.1. Also shown are the corresponding analytic solutions given by equation (1.2). The agreement between the two is good.

1.4.4 Limitations of the Numerical Method

The *numerical method* described here generally performs very well; however, it has a couple of failure scenarios that are important to understand (and provide the basis for understanding GYRE's failure modes). These scenarios arise through poor choices of the spatial grid used to discretize the wave equation, and/or the frequency grid used to search for roots of the discriminant function.

Insufficient Spatial Resolution

The cost of evaluating the determinant of the system matrix scales proportionally to the number of grid points N used for the *discretization*. Therefore, in the interests of computational efficiency, we want to make N as small as possible.

However, things go wrong when N becomes too small. Fig. 1.4 demonstrates this by plotting the discriminant function for the stretched-string BVP with $N = 7$. Compared against Fig. 1.1, we see that toward larger σ the roots of the discriminant function become progessively shifted toward lower frequencies; and, above $\sigma \approx 3.5\pi c/L$, they disappear altogether.

Fig. 1.4: Plot of the discriminant function (σ) as a function of the frequency σ, for the stretched-string BVP with $N = 7$. The orange dots highlight where $= 0$. The function has been scaled so that $(0) = 1$. (`Source`)

To understand this behavior, recall that the detemrinant of an $N \times N$ matrix can be expressed (via Laplace expansion) as the sum of *N* terms; and each term itself involves the product of N matrix elements, picked so that each row/column is used only once in the construction of the term. With these points in mind, we can see from the definition (1.3) of that its determinant (i.e., the discriminant function) must be a polynomial in σ^2 of order $N - 2$; and as such, it can have at most $N - 2$ (in this case, 5) roots. This leads us to important lesson #1:

Attention: The number of points adopted in the discretization limits the number of modes that can be found. With a spatial grid of* N points, there are only (of order) N distinct numerical solutions.

Fig. 1.5: Plot of the eigenfunctions $\tilde{y}$ as a function of spatial coordinate x, for the first three modes of the stretched-string BVP with $N = 7$. The discrete points show the numerical functions, and the solid lines the corresponding analytic functions. (`Source`)

Returning to Fig. 1.4, the shift in eigenfrequency for the modes that *are* found occurs due to inadequate resolution of the eigenfunctions. We can see this in Fig. 1.5, which reprises Fig. 1.3 for $N = 7$. Clearly, the spatial oscillations of the modes are poorly resolved; the $n = 3$ mode, for instance, is sampled with only one point per quarter wavelength. It's little wonder that the corresponding eigenfrequencies are off. This brings us to important lesson #2 (closely related to #1):

Attention: The spatial resolution adopted in the discretization determines the accuracy of the modes found. A given eigenfrequency will be accurate only when the spatial grid spacing is appreciably smaller than the spatial variation scale of the corresponding eigenfunction.

Insufficient Frequency Resolution

When searching for root brackets, we have to evaluate the discriminant function a total of M times. Therefore, as with N, computational efficiency dictates that we want to make M as small as possible. Again, however, things go wrong if M is too small. Fig. 1.6 reprises Fig. 1.2, but adopting a much coarser frequency grid with only $M = 4$ points.

Fig. 1.6: Plot of the discriminant values $\{\}$ on the discrete frequency grid $\{\sigma\}$, for the stretched-string BVP with $N = 50$ and $M = 4$. The orange halos indicate adjacent points that bracket a root $= 0$. (`Source`)

Clearly, all but the lowest-frequency ($n = 1$) mode are missed in the bracketing process. This is admittely an extreme example, but nicely demonstrates the consequences of too coarse a frequency grid, and gives us important lesson #3:

Attention: The frequency resolution adopted in the root bracketing influences the completeness of the modes found. All modes will be found only when the frequency grid spacing is smaller than the eigenfrequency separation of adjacent modes.

1.4.5 From Stretched String to GYRE

The numerical technique used in this chapter to solve the stretched-string BVP provides a strong analog to how GYRE solves the oscillation equations. The full, nasty details of GYRE's approach are laid out in XXXX

In this section, let's review the similarities and differences between the two.

Similarities

GYRE shares the following similarities to

- Separation of variables
- Discretization
- Root bracketing for adiabatic
- Root refinement
- Failure scenarios

Differences in GYRE

- More variables
- Non-uniform spatial and frequency grids
- The stretched string is discretized using a second-order form of the wave equation. GYRE breaks this up into 1st-order forms
- The discretizaton scheme is more complex

- Complex root solving for non-adiabatic

1.5 Working with Grids

This chapter describes how GYRE sets up its frequency and spatial grids, and discusses strategies for ensuring these grids are optimal.

1.5.1 Frequency Grids

GYRE searches for sign changes of the discriminant function (ω) on a grid $\{\omega_1, \omega_2, \ldots, \omega_M\}$ in the dimensionless frequency $\omega \equiv \sqrt{R^3/GM}\sigma$. The computational cost of a calculation scales with the total number of points M in this grid, while the grid's resolution — i.e., the spacing between adjacent points — impacts the completeness of the modes found by GYRE (see the *Limitations of the Numerical Method* section for a discussion of these behaviors in the context of the stretched string BVP).

GYRE constructs a fresh frequency grid for each combination of harmonic degree ℓ and azimuthal order m specified in the `&mode` namelist groups (see the *Namelist Input Files* chapter for more details).

The starting

point for each of these grids is the *scaffold grid*, which comprises the following:

- an inner point $x =$;
- an outer point $x =$;
- the subset of points of the input model grid satisfying $< x <$

By default, and are obtained from the input model grid as well, meaning that the scaffold grid is identical to the model grid. However, either or both can be overridden using the `x_i` and `x_o` parameters, respectively, of the `&grid` namelist group.

The grid is established

1.5.2 Spatial Grids

GYRE discretizes the oscillation equations on a spatial grid $\{x_1, x_2, \ldots, x_N\}$ in the dimensionless radial coordinate $x \equiv r/R$. The computational cost of a calculation scales with the total number of points N in this grid, while the grid's resolution — i.e., the spacing between adjacent points — impacts both the number of modes that can be found by GYRE, and the accuracy of these modes (see the *Limitations of the Numerical Method* section for a discussion of these behaviors in the context of the stretched string BVP).

Scaffold Grid

GYRE constructs a fresh spatial grid for each combination of harmonic degree ℓ and azimuthal order m specified in the `&mode` namelist groups (see the *Namelist Input Files* chapter for more details). The starting point for each of these grids is the *scaffold grid*, which comprises the following:

- an inner point $x =$;
- an outer point $x =$;
- the subset of points of the input model grid satisfying $< x <$

By default, and are obtained from the input model grid as well, meaning that the scaffold grid is identical to the model grid. However, either or both can be overridden using the `x_i` and `x_o` parameters, respectively, of the `&grid` namelist group.

Iterative Refinement

GYRE refines a scaffold grid through a sequence of iterations. During a given iteration, each subinterval $[x_k, x_{k+1}]$ ($k = 1, 2, \ldots, N-1$) is assessed against various criteria (discusssed in greater detail below). If any criteria match, then the subinterval is refined by bisection, inserting an additional point at the midpoint

$$x_{k+1/2} = \frac{x_k + x_{k+1}}{2}.$$

The sequence terminates if no refinements occur during a given iteration, or if the number of completed iterations equals the value specified by the `n_iter_max` parameter of the `&grid` namelist group.

Mechanical Criterion

The wave criterion involves a local analysis of the mechanical parts of the oscillation equations, with the goal of improving resolution where the displacement perturbation is rapidly varying. Within the subinterval $[x_k, x_{k+1}]$, the y_1 and y_2 solutions (see the *Mathmatical Formalism* chapter) take the approximate form

$$y_{1,2}(x) \sim \exp[\chi \, \ln x],$$

where χ is one of the eigenvalues of the mechanical (upper-left) 2×2 submatrix of the full Jacobian matrix , evaluated at the midpoint $x_{k+1/2}$.

In propagation zones the imaginary part χ_i of the eigenvalue gives the local wavenumber in $\ln x$ space, and $2\pi\chi_\mathrm{i}^{-1}$ the corresponding wavelength; while in evanescent zones the real part χ_r gives the local exponential growth/decay rate, and χ_r^{-1} the corresponding e-folding length.

Based on this analysis, the criterion for refinement of the subinterval is

$$(\ln x_{k+1} - \ln x_k) \, \max(\alpha_\mathrm{osc}|\chi_\mathrm{i}|, \alpha_\mathrm{exp}|\chi_\mathrm{r}|) > 2\pi$$

This causes refinement if the subinterval width (in $\ln x$ space) exceeds α_osc^{-1} times the local wavelength, or $2\pi\alpha_\mathrm{exp}^{-1}$ times the local e-folding length. The controls α_exp and α_exp are set via the `alpha_exp` and `alpha_osc` parameters, respectively, of the `&grid` namelist group.

Tip: While `alpha_exp` and `alpha_osc` default to zero, it is highly recommended to use non-zero values for these parameters, to ensure adequate resolution of solutions throughout the star. Reasonable starting choices are `alpha_osc = 10` and `alpha_exp = 2`.

Because there are two possible values for χ, the above refinement criterion is applied twice (once for each). Moreover, because χ depends implicitly on the oscillation frequency, the criterion is applied for each frequency in the grid $\{\omega_1, \omega_2, \ldots, \omega_M\}$.

Thermal Criterion

Similar to the wave criterion discussed above, the thermal criterion involves a local analysis of the energetic parts of the oscillation equation, with the goal of improving resolution where the thermal timescale is very long and perturbations are almost adiabatic. Within the subinterval $[x_k, x_{k+1}]$, the y_5 and y_6 perturbation take the approximate form

$$y_{5,6}(x) \sim \exp[\pm\tau \, (\ln x - \ln x_{k+1/2})],$$

where $\pm\tau$ are the eigenvalues of the matrix formed from the energetic (bottom-rright) 2×2 submatrix of the full Jacobian matrix , evaluated at the midpoint $x_{k+1/2}$.

Based on this analysis, the criterion for refinement of the subinterval is

$$(\ln x_{k+1} - \ln x_k)\, \alpha_{\rm thm} |\tau| > 1.$$

The control $\alpha_{\rm thm}$ is set via the `alpha_thm` parameters of the `&grid` namelist group.

Because τ depends implicitly on the oscillation frequency, this criterion is applied for each frequency in the grid $\{\omega_1, \omega_2, \ldots, \omega_M\}$.

Structural Criteria

The structural criteria have the goal of improving resolution where the stellar structure coefficients are changing rapidly. For a given coefficient C, the criterion for refinement of the subinterval $[x_k, x_{k+1}]$ is

$$(\ln x_{k+1} - \ln x_k)\, \alpha_{\rm str}\, |\ln C \ln x| > 1$$

The control $\alpha_{\rm thm}$ is set via the `alpha_thm` parameter of the `&grid` namelist group. This criterion is applied to the $V_2 \equiv V/x$, U, A^*, c_1 and Γ_1 coefficients (see the structure-coeffs section).

Central Criteria

All of the above criteria depend on the logarithmic subinterval width $(\ln x_{k+1} - \ln x_k)$, and cannot be applied to the first subinterval $[x_1, x_2]$ if it extends to the center of the star $x = 0$. In such cases, the `resolve_ctr` parameter of the `&grid` namelist group determines whether the subinterval is refined. If set to `.FALSE.`, then no refinement occurs; while if set to `.TRUE.`, then the refinement criteria are

$$\chi_{\rm i} > 0$$

or

$$\alpha_{\rm ctr} |\chi_{\rm r}| > 1$$

where χ is the eigenvalue from the local analysis (see the *Mechanical Criterion* section) corresponding to the solution that remains well-behaved at the origin. The first criterion causes refinement if the subinterval is in a propagation zone, and the second if the solution slope $|\ln y \ln x| \sim |\chi_{\rm r}|$ exceeds $\alpha_{\rm ctr}^{-1}$. The control $\alpha_{\rm ctr}$ is set via the `alpha_ctr` parameter of the `&grid` namelist group.

Tip: While `alpha_ctr` defaults to zero, it is highly recommended to use a non-zero value for this parameter, to ensure adequate resolution of solutions at the center. A reasonable starting choice is `alpha_ctr = 10`.

Because χ depends implicitly on the oscillation frequency, these criteria are applied for each frequency in the grid $\{\omega_1, \omega_2, \ldots, \omega_M\}$.

Limiting Controls

A couple of additional controls affect the iterative refinement described above. Refinement of the $[x_k, x_{k+1}]$ subinterval *always* occurs if

$$x_{k+1} - x_k > \Delta x_{\rm max},$$

and *never* occurs if

$$x_{k+1} - x_k < \Delta x_{\min}.$$

The $\Delta x_{\max}$ and $\Delta x_{\max}$ controls are set by the `dx_max` and `dx_min` parameters, respectively, of the `&grid` namelist group.

1.6 Interpreting Output Files

1.6.1 TXT-Format Files

1.6.2 HDF-Format Files

1.6.3 Python Support

1.7 Installation

This chapter discusses GYRE installation in detail. If you just want to get up and running, have a look at the *Quick Start* chapter.

1.7.1 Pre-Requisites

To compile and run GYRE, you'll need the following software components:

- A modern (2003+) Fortran compiler
- The BLAS linear algebra library
- The LAPACK linear algebra library
- The LAPACK95 Fortran 95 interface to LAPACK
- The HDF5 data management library
- The crlibm correctly rounded math library
- The crmath Fortrran 2003 interface to crlibm
- An OpenMP-aware version of the ODEPACK differential equation library (optional)

On Linux and MacOS platforms, these components are bundled together in the MESA Software Development Kit (SDK), which can be downloaded from the MESA SDK homepage. Using this SDK is strongly recommended.

1.7.2 Building GYRE

Download

Download the GYRE source code, and unpack it from the command line using the **`tar`** utility:

Set the `GYRE_DIR` environment variable with the path to the newly created source directory; this can be achieved e.g. using the **`dirname`** built-in command:

Compile

Compile GYRE using the **make** utility:

(the `-j` flags tells **make** to use multiple cores, speeding up the build).

Test

o check that GYRE has compiled correctly and gives reasonable results, you can run the calculation test suite via the command

The initial output from the tests should look something like this:

```
TEST numerics (OpenMP)...
 ...succeeded
TEST numerics (band matrix)...
 ...succeeded
TEST numerics (*_DELTA frequency units)...
 ...succeeded
TEST numerics (rotation, Doppler shift)...
 ...succeeded
TEST numerics (rotation, traditional approximation)...
 ...succeeded
```

If things go awry, consult the *Troubleshooting* chapter.

1.7.3 Custom Builds

Custom builds of GYRE can be created by setting certain environment variables, and/or variables in the file *$GYRE_DIR*`/src/build/Makefile`, to the value `yes`. The following variables are currently supported:

DEBUG Enable debugging mode (default `no`)

OMP Enable OpenMP parallellization (default `yes`)

MPI Enable MPI parallelizatiomn (default `no`)

DOUBLE_PRECISION Use double precision floating point arithmetic (default `yes`)

CRMATH Use correctly rounded math functions (default `yes`)

IEEE Use Fortran IEEE floating point features (default `no`)

FPE Enable floating point exception checks (default `yes`)

HDF5 Include HDF5 support (default `yes`)

EXPERIMENTAL Enable experimental features (default `no`)

If a variable is not set, then its default value is assumed.

1.7.4 Git Access

Sometimes, you'll want to try out new features in GYRE that haven't yet made it into a formal release. In such cases, you can check out GYRE directly from the rhdtownsend/gyre git repository on GitHub:

However, a word of caution: GYRE is under constant development, and features in the main (`master`) branch can change without warning.

1.8 Namelist Input Files

GYRE reads parameters from an input file which defines a number of Fortran namelist groups, as described below.

1.8.1 Constants

The `&constants` namelist group defines various physical constants; the input file should contain exactly one. Allowable parameters are:

`G_GRAVITY` Gravitational constant G

`C_LIGHT` Speed of light *in vacuo* c

`A_RADIATION` Radiation constant a

`M_SUN` Solar mass $M_\odot$

`R_SUN` Solar radius $R_\odot$

`L_SUN` Solar luminosity $L_\odot$

`GYRE_DIR` Top-level GYRE directory; overrides the `GYRE_DIR` environment variable

All of these constants are in cgs units (where applicable), and the default values are defined in src/common/gyre_constants.fpp.

1.8.2 Stellar Model Parameters

The `&model` namelist group defines stellar model parameters; the input file should contain exactly one. Allowable parameters are:

`model_type` Type of model to use; one of:

- `'HOM'` : Homogeneous compressible model
- `'POLY'` : Polytropic model read from external file
- `'EVOL'` : Evolutionary model read from external file

`file` Name of file, when `model_type` is `'POLY'` or `'EVOL'`

`file_format` Format of file, when `model_type` is `'EVOL'`; one of

- `'AMDL'` : AMDL-format binary file
- `'B3'` : B3-format HDF5 file
- `'FAMDL'` : FAMDL-format text file
- `'FGONG'` : FGONG-format text file
- `'GSM'` : GSM-format HDF5 file
- `'LOSC'` : LOSC-format text file
- `'MESA'` : MESA GYRE-format text file
- `'OSC'` : OSC-format text file
- `'WDEC'` : WDEC-format text file

`data_format` (default `''`, indicates auto-select) Fortran format specifier for data read from OSC-, FGONG- and FAMDL-format files

deriv_type (default 'MONO') Cubic interpolation derivatives type, when model_type is 'POLY' or 'EVOL'; one of

- 'NATURAL' : Natural (spline) derivatives
- 'FINDIFF' : Finite-difference derivatives
- 'MONO' : Monotonized derivatives (default)

Gamma_1 (default 5/3) First adiabatic exponent, when model_type is 'HOM'

grid_type (default 'UNI') Model grid type, when model_type is 'HOM'; one of

- 'UNI' : Uniform spacing
- 'GEO' : Geometric spacing
- 'LOG' : Logarithmic spacing

n (default 10) Number of points in model grid, when model_type is 'HOM'

s (default 1) Skewness parameter for model grid, when model_type is 'HOM' and grid_type is 'GEO' or 'LOG'

x_i (default 0) Inner boundary coordinate of model grid, when model_type is 'HOM'

x_o (default 1) Outer boundary coordinate of model grid, when model_type is 'HOM'

uniform_rot (default .FALSE.) Flag to force uniform rotation

Omega_rot (default .FALSE.) Rotation angular velocity, when uniform_rot is .TRUE.

Omega_units (default 'NONE') Units of Omega_rot; one of

- 'NONE' : Dimensionless angular frequency
- 'HZ' : Linear frequency in Hz[1]
- 'UHZ' : Linear frequency in μHz[1]
- 'RAD_PER_SEC' : Angular frequency in radians per second[1]
- 'CYC_PER_DAY' : Linear frequency in cycles per day[1]
- 'CRITICAL' : Fraction of the Roche critical rate[1]

dx_snap (default 0) Threshold for snapping model points together, when model_type is 'EVOL'. If a pair of points are separated by less than dx_snap, they are snapped together.

add_center (default .TRUE.) Flag to add a center point to the model, when model_type is 'EVOL' or 'POLY'. If a point does not already exist at the origin, then one is added

repair_As (default .FALSE.) Flag to repair inaccuracies in the dimensionless Brunt-Väisälä frequency at density discontinuities

1.8.3 Mode Parameters

The &mode namelist group defines mode parameters; the input file can contain one or more. Allowable parameters are:

l (default 0) Harmonic degree ℓ

m (default 0) Azimuthal order m

tag Tag for controlling selection of other parameters

[1] This option is only available when model_type is 'EVOL'

n_pg_min (default -HUGE) Filter for minimum radial order

n_pg_max (default +HUGE) filter for maximum radial order

rossby (default .FALSE.) Flag to use Rossby-mode angular eigenvalues/eigenfunctions

static (default .FALSE.) Flag to solve for the static ($\omega \to 0$) limit

1.8.4 Oscillation Parameters

The `&osc` namelist group defines oscillation parameters; the input file can contain one or more, but only the last (matching) one is used. Allowable parameters are:

inner_bound (default 'REGULAR') Inner boundary conditions; one of:

- `'REGULAR'` : Regularity-enforcing (only valid when inner grid point is at $x = 0$)
- `'ZERO_R'` : Zero radial displacement (only valid when inner grid point is at $x \neq 0$)
- `'ZERO_H'` : Zero horizontal displacement (only valid when inner grid point is at $x \neq 0$)

outer_bound (default 'VACUUM') Outer boundary conditions; one of:

- `'VACUUM'` : Zero surface pressure
- `'DZIEM'` : Formulation following Dziembowski (1971)
- `'UNNO'` : Formulation following Unno et al. (1989)
- `'JCD'` : Formulation following Jörgen Christensen-Dalsgaard (ADIPLS)

variables_set (default 'GYRE') Dependent variables in oscillation equations; one of:

- `'GYRE'` : GYRE formulation, as desciribed in the equations.pdf document
- `'DZIEM'` : Formulation following Dziembowski (1971)
- `'JCD'` : Formulation following Jörgen Christensen-Dalsgaard (ADIPLS)
- `'MIX'` : Mixed formulation (`'JCD'` for gravitational components, `'DZIEM'` for mechanical components)
- `'LAGP'` : Lagrangian pressure perturbation formulation

alpha_gr (default 1.) Scaling factor for gravitational potential perturbations (see $\alpha_{\rm gr}$ term in the osc-equation section). Set to `0.` to implement the Cowling approximation

alpha_th (defaualt 1.) Scaling factor for the thermal timescale (see $\alpha_{\rm th}$ term in the osc-equations section). Set to `0.` to implement the non-adiabatic reversible (NAR) approximation, and to large values to approach the adiabatic limit

alpha_hf (defaualt 1.) Scaling factor for horizontal flux perturbations (see $\alpha_{\rm hf}$ term in the osc-equations section). Set to `0.` to implement the non-adiabatic radial flux (NARF) approximation

inertia_norm (default 'BOTH') Inertia normalization factor; one of

- `'RADIAL'` : Radial amplitude squared, $|\xi_{\rm r}|^2$, evaluated at `x_ref`
- `'HORIZ'` : Horizontal amplitude squared, $|\lambda||\xi_{\rm h}|^2$, evaluated at `x_ref`
- `'BOTH'` : Overall amplitude squared, $|\xi_{\rm r}|^2 + |\lambda||\xi_{\rm h}|^2$, evaluated at `x_ref`

time_factor (default 'OSC') Time-dependence factor in pulsation equations; one of:

- `OSC` : Oscillatory, $\propto \exp(-{\rm i}\omega t)$
- `EXP` : Exponential, $\propto \exp(-\omega t)$

conv_scheme (default 'FROZEN_PESNELL_1'`) Scheme for treating convection; one of:

- 'FROZEN_PESNELL_1' : Freeze convective heating altogether; case 1 described by Pesnell (1990)
- 'FROZEN_PESNELL_4' : Freeze Lagrangian perturbation of convective luminosity; case 4 described by Pesnell (1990)

zeta_scheme (default PESNELL) Scheme for evaluating frequency weight function ζx and integral dimensionless eigenfrequency $\omega_{\rm int}$; one of:

- 'PESNELL' : Evaluate using eqn. (A5) of Pesnell (1987)
- 'DUPRET' : Evaluate using eqn. (1.71) of Dupret (2002, PhD thesis)
- 'KAWALER' : Evaluate using eqn. (7) of Kawaler et al. (1985)
- 'KAWALER_GRAV': Evaluate using the g-mode part in eqn. (7) of Kawaler et al. (1985)

deps_scheme (default 'MODEL') Scheme for calculating burning partial derivatives $(\partial \ln \epsilon/\partial \ln T)_\rho$ and $(\partial \ln \epsilon/\partial \ln \rho)_T$; one of

- 'MODEL' : Use values from model
- 'FILE' : Use complex (phase-lagged) values from separate file

deps_file (default '') Name of epsilon partial derivatives file, when deps_scheme is 'FILE'

deps_file_format (default 'WOLF') Format of epsilon partial derivative file, when deps_scheme is 'FILE'; one of:

- 'WOLF' : Format used in preparation of Wolf et al. (2018)

x_ref (default 1 or outer grid point, whichever is smaller) Reference fractional radius for photosphere, normalizations etc.

nonadiabatic (default .FALSE.) Flag to include non-adiabatic effects

quasiad_eigfuncs (default .FALSE.) Flag to calculate quasi-adiabatic entropy/luminosity eigenfunctions during adiabatic calculations

eddington_approx (default .FALSE.) Flag to use the Eddington approximation

reduce_order (default .TRUE.) Flag to reduce the order of the *adiabatic* radial-pulsation equations from 4 to 2

tag_list (default '', which matches all) Comma-separated list of &mode tags to match

1.8.5 Rotation Parameters

The &rot namelist group defines rotational parameters; the input file can contain one or more, but only the last (matching) one is used. Allowable parameters are:

coriolis_method (default 'NULL') Method used to treat the Coriolis force; one of:

- 'NULL' : Neglect the Coriolis force
- 'TAR' : Use the traditional approximation of rotation

Omega_rot_source (default 'MODEL') Sourcce for rotational angular velocity; one of:

- 'MODEL' : Use data from the stellar model
- 'UNIFORM' : Uniform rotation, with angular velocity set by Omega_rot parameter

Omega_rot (default 0) Rotation angular frequency (when Omega_rot_source is 'UNIFORM')

Omega_rot_units (default 'NULL') Units of Omega_rot (when Omega_rot_source is 'UNIFORM'); one of:

- `'NONE'` : Dimensionless angular frequency
- `'HZ'` : Linear frequency in Hz[1]
- `'UHZ'` : Linear frequency in μHz[1]
- `'RAD_PER_SEC'` : Angular frequency in radians per second[1]
- `'CYC_PER_DAY'` : Linear frequency in cycles per day[1]
- `'CRITICAL'` : Fraction of the Roche critical rate[1]

`complex_lambda` (default `.FALSE.`) Flag to use complex arithmetic when evaluating the TAR angular eigenvalue λ

`tag_list` (default `''`, which matches all) Comma-separated list of `&mode` tags to match

1.8.6 Numerical Parameters

The `&num` namelist group defines numerical method parameters; the input file can contain one or more, but only the last (matching) one is used. Allowable fields are:

`diff_scheme` (default `'COLLOC_GL2'`) Difference equation scheme; one of:

- `'COLLOC_GL2'` : Second-order Gauss-Legendre collocation
- `'COLLOC_GL4'` : Fourth-order Gauss-Legendre collocation
- `'COLLOC_GL6'` : Sixth-order Gauss-Legendre collocation
- `'MAGNUS_GL2'` : Second-order Gauss-Legendre Magnus
- `'MAGNUS_GL4'` : Sourth-order Gauss-Legendre Magnus
- `'MAGNUS_GL6'` : Sixth-order Gauss-Legendre Magnus
- `'MIRK'` : Fourth-order mono-implicit Runge-Kutta (experimental)
- `'TRAPZ'` : Trapezoidal, with the prescription by Sugimoto (1970) for non-adiabatic cases

`r_root_solver` (default `'BRENT'`) Root solver for real arithmetic; one of:

- `'BRENT'` : Brent's method

`c_root_solver` (default `'RIDDERS'`) Root solver for complex arithmetic; one of

- `'RIDDERS'` : Complex Ridders' method
- `'SECANT'` : Secant method
- `'SIMPLEX'` : Simplex method

`n_iter_max` (default `50`) Maximum number of iterations in root-finding algorithm

`matrix_type` (default `'BLOCK'`) Storage type of system matrix; one of

- `'BAND'` : Band-structured
- `'BLOCK'` : Block-structued

`deflate_roots` (default `.TRUE.`) Flag to use root deflation, which can avoid the same eigenfrequency being found multiple times

`restrict_roots` (default `.TRUE.`) Flag to check each roots found lies within the bounds of the frequency scan

`tag_list` (default `''`, which matches all) Comma-separated list of `&mode` tags to match

[1] This option is only available when `model_type` is `'EVOL'`

1.8.7 Frequency Scan Parameters

The `&scan` namelist group(s) defines frequency scan parameters; the input file can contain one or more, and points defined by each `&scan` namelist group are merged together. Allowable parameters are:

`grid_type` (default `'LINEAR'`) Distribution of frequency points; one of:

- `'LINEAR'` : Uniform in frequency
- `'INVERSE'` : Uniform in inverse frequency (i.e., period)

`grid_frame` (default `'INERTIAL'`) Frame in which `grid_type` applies; one of:

- `'INERTIAL'` : Inertial frame
- `'COROT_I'` : Co-rotating frame at inner boundary
- `'COROT_O'` : Co-rotating frame at outer boundary

`freq_min` (default `1`) Minimum frequency

`freq_max` (default `10`) Maximum frequency

`n_freq` (default `10`) Number of frequency points

`freq_min_units` (default `'NONE'`) Units of `freq_min`; one of:

- `'NONE'` : Dimensionless angular frequency
- `'HZ'` : linear frequency in Hz[1]
- `'UHZ'` : linear frequency in μHz[1]
- `'RAD_PER_SEC'` : angular frequency in radians per second[1]
- `'CYC_PER_DAY'` : linear frequency in cycles per day[1]
- `'ACOUSTIC_DELTA'` : Fraction of the asymptotic acoustic large frequency separation $\Delta\nu$
- `'GRAVITY_DELTA'` : Fraction of the asymptotic inverse gravity period separation $(\Delta P)^{-1}$
- `'UPPER_DELTA'` : Greater of $\Delta\nu$ and $(\Delta P)^{-1}$
- `'LOWER_DELTA'` : Lesser of $\Delta\nu$ and $(\Delta P)^{-1}$
- `'ACOUSTIC_CUTOFF'` : fraction of the acoustic cutoff frequency[1]
- `'GRAVITY_CUTOFF'` : fraction of the gravity cutoff frequency[1]
- `'ROSSBY_I'` : fraction of Rossby frequency at inner boundary
- `'ROSSBY_O'` : fraction of Rossby frequency at outer boundary

`freq_max_units` (default `'NONE'`) Units of `freq_max`; same options as `freq_min_units`

`freq_min_frame` (default `'INERTIAL'`) Frame of `freq_min`; one of:

- `'INERTIAL'` : Inertial frame
- `'COROT_I'` : Co-rotating frame at inner boundary
- `'COROT_O'` : Co-rotating frame at outer boundary

`freq_max_frame` (default `'INERTIAL'`) Frame of `freq_max`; same options as `freq_min_frame`

`tag_list` (default `''`, which matches all) Comma-separated list of `&mode` tags to match

[1] This option is only available when `model_type` is `'EVOL'`

1.8.8 Grid Parameters

The `&grid` namelist group defines the parameters used to set up the calculation grid; the input file can contain one or more, but only the last (matching) one is used. Allowable parameters are:

`x_i` (default based on model grid) Inner boundary coordinate of calculation grid

`x_o` (default based on model grid) Outer boundary coordinate of calculation grid

`alpha_osc` (default `0`) Oscillatory resampling parameter $\alpha_{\rm osc}$

`alpha_exp` (default `0`) Exponential resampling parameter $\alpha_{\rm exp}$

`alpha_thm` (default `0`) Thermal resampling parameter $\alpha_{\rm thm}$

`alpha_str` (default `0`) Structural resampling parameter $\alpha_{\rm str}$

`dx_min` (default `SQRT(EPSILON(1._WP))`) Minimum spacing of grid points

`n_inner` (default `0`) Number of extra point to insert at center

`n_floor` (default `0`) Minimum total number of grid points

`n_iter_max` (default `8`) Maximum number of refinement iterations

`tag_list` (default `''`, which matches all) Comma-separated list of `&mode` tags to match

See Understanding Grids for further details, in particular a discussion of how the `alpha_*` resampling parameters work.

1.8.9 Output Parameters

The `&ad_output` and `&nad_output` namelist groups determine the output produced at the end of a run, from the adiabatic and non-adiabatic calculation stages, respectively; the input file should contain exactly one of each. Allowable parameters are:

`summary_file` (default `''`) Name of summary file

`summary_file_format` (default `'HDF'`) Format of summary file; one of

`'HDF'` : HDF5 file `'TXT'` : Text file

`summary_item_list` (default `'l,n_pg,omega,freq'`) Comma-separated list of output items to write to summary file; see the *Summary Files* section for possible choices

`summary_filter_list` (default `''`) Comma-separated list of filtering criteria for summary files; see the output-filtering section for possible choices

`detail_template` (default `''`) Name template of detail files. Names are generated using the following pattern substitutions:

- `'%J'` : Unique mode index j, formatted in fixed-width field
- `'%j'` : Same as `'%J'`, but formatted in variable-width field
- `'%L'` : Harmonic degree ℓ, formatted in fixed-width field
- `'%l'` : Same as `'%L'`, but formatted in variable-width field
- `'%N'` : Radial order $n_{\rm pg}$, formatted in fixed-width field
- `'%n'` : Same as `'%N'`, but formatted in variable-width field
- `'%P'` : Acoustic wave winding number $n_{\rm p}$, formatted in fixed-width field
- `'%p'` : Same as `'%P'`, but formatted in variable-width field

- `'%G'` : Gravity wave winding number $n_{\rm g}$, formatted in fixed-width field
- `'%g'` : Same as `'%G'`, but formatted in variable-width field

`detail_file_format` (default `'HDF'`) Format of detail files; one of

- `'HDF'` : HDF5 file
- `'TXT'` : text file

`detail_item_list` (default `'l,n_pg,omega,freq,x,xi_r,xi_h'`) Comma-separated list of output items to write to detail files; see the *Detail Files* section for possible choices

`detail_filter_list` (default `''`) Comma-separated list of filtering criteria for detail files; see the output-filtering section for possible choices

`freq_units` (default `NONE`) Units of `freq` output item; one of:

- `'NONE'` : Dimensionless angular frequency
- `'HZ'` : linear frequency in Hz[1]
- `'UHZ'` : linear frequency in μHz[1]
- `'RAD_PER_SEC'` : angular frequency in radians per second[1]
- `'CYC_PER_DAY'` : linear frequency in cycles per day[1]
- `'ACOUSTIC_DELTA'` : Fraction of the asymptotic acoustic large frequency separation $\Delta\nu$
- `'GRAVITY_DELTA'` : Fraction of the asymptotic inverse gravity period separation $(\Delta P)^{-1}$
- `'UPPER_DELTA'` : Greater of $\Delta\nu$ and $(\Delta P)^{-1}$
- `'LOWER_DELTA'` : Lesser of $\Delta\nu$ and $(\Delta P)^{-1}$
- `'ACOUSTIC_CUTOFF'` : fraction of the acoustic cutoff frequency[1]
- `'GRAVITY_CUTOFF'` : fraction of the gravity cutoff frequency[1]
- `'ROSSBY_I'` : fraction of Rossby frequency at inner boundary
- `'ROSSBY_O'` : fraction of Rossby frequency at outer boundary

`freq_frame` (default `INERTIAL`) Frame of `freq` output item; one of:

- `'INERTIAL'` : Inertial frame
- `'COROT_I'` : Co-rotating frame at inner boundary
- `'COROT_O'` : Co-rotating frame at outer boundary

`label` (default `''`) Textual label to add to all output files

1.9 Output Files

The output files produced by GYRE fall into two categories. *Summary files* gather together information about all modes found during a GYRE run – for instance, eigenfrequencies, radial orders, etc. *Detail files*, by contrast, gather together information about a single mode – for instance, eigenfunctions, rotation kernel, etc. For both categories, the file format on disk can be either text-based or HDF-based.

[1] This option is only available when `model_type` is `'EVOL'`

1.9.1 Summary Files

A summary file gathers together information about all modes found during a GYRE run. The data written to summary files is controlled by the `summary_item_list` parameter of the `&ad_output` namelist group (for adiabatic calculations) and the `&nad_output` namelist group (for nonadiabatic calculations). This parameter is a comma-separated list of items to appear in the summary file. The items come in two flavors:

- *scalar* items comprise a single value, typically pertaining to all modes (i.e., a global quantity)
- *array* items comprise a sequence of values, with each value pertaining to a single mode. The sequence follows the same order in which modes were found during the GYRE run.

The following subsections describe the items that may appear in a `summary_item_list` parameter, grouped together by functional area.

Solution Data

`omega` (complex array) Dimensionless eigenfrequency ω

Observables

`freq` (complex array) Dimensioned eigenfrequency. The units and reference frame are controlled by `freq_units` and `freq_frame` parameters of the `&ad_output` and `&nad_output` namelist groups

`freq_units` (character scalar) Units of `freq`

`freq_frame` (character scalar) Reference frame of `freq`

`f_T` (real array) Effective temperature perturbation amplitude $f_{\rm T}$. Evaluated using eqn. 5 of Dupret et al. (2003)

`f_g` (real array) Effective gravity perturbation amplitude $f_{\rm g}$. Evaluated using eqn. 6 of Dupret et al. (2003)

`psi_T` (real array) Effective temperature perturbation phase $\psi_{\rm T}$. Evaluated using eqn. 5 of Dupret et al. (2003)

`psi_g` (real array) Effective gravity perturbation phase $\psi_{\rm g}$

Classification & Validation

`j` (integer array) Unique mode index j. The first mode found during the GYRE run has $j = 1$, the second $j = 2$, and so on

`l` (integer array) Harmonic degree ℓ

`l_i` (complex array) Effective harmonic degree at inner boundary $\ell_{\rm i}$

`m` (integer array) Azimuthal order m

`n_p` (integer array) Acoustic-wave winding number $n_{\rm p}$

`n_g` (integer array) Gravity-wave winding number $n_{\rm g}$

`n_pg` (integer array) Radial order $n_{\rm pg}$ within the Eckart-Scuflaire-Osaki-Takata scheme (see Takata, 2006)

`omega_int` (complex array) Dimensionless eigenfrequency ω from integral expression. Evaluated using eqn. 1.71 of Marc-Antoine Dupret's PhD thesis

Perturbations

x_ref (real array) Fractional radius of reference location $x_{\rm ref}$

xi_r_ref (complex array) Radial displacement perturbation $\xi_{\rm r}$ at reference location $x_{\rm ref}$, in units of R

xi_h_ref (complex array) Horizontal displacement perturbation $\xi_{\rm h}$ at reference location $x_{\rm ref}$, in units of R

eul_phi_ref (complex array) Eulerian potential perturbation Φ' at reference location $x_{\rm ref}$, in units of GM/R

deul_phi_ref (complex array) Eulerian potential gradient perturbation $\mathrm{d}\Phi'/\mathrm{d}x$ at reference location $x_{\rm ref}$, in units of GM/R^2

lag_S_ref (complex array) Lagrangian specific entropy perturbation δS at reference location $x_{\rm ref}$, in units of c_P

lag_L_ref (complex array) Lagrangian radiative luminosity perturbation $\delta L_{r,\rm R}$ at reference location $x_{\rm ref}$, in units of L

Energetics & Transport

eta (real array) Normalized growth rate η. Evaluated using expression in text of page 1186 of Stellingwerf (1978)

E [(real array)] Mode inertia E, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$

E_p (real array) Acoustic inertia $E_{\rm p}$, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$ in acoustic-wave propagation regions

E_p (real array) Gravity inertia $E_{\rm g}$, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$ in gravity-wave propagation regions

E_norm (real array) Normalized inertia $E_{\rm norm}$. The normalization is controlled by the `inertia_norm` parameter of the `&osc` namelist group

E_ratio (real array) Ratio of mode inertia inside/outside the reference location $x_{\rm ref}$

H (real array) Mode energy H, in units of GM^2/R

W (real array) Mode work W, in units of GM^2/R. Evaluated by integrating $\mathrm{d}W/\mathrm{d}x$

W_eps (real array) Mode nuclear work W_ϵ, in units of GM^2/R. Evaluated by integrating $\mathrm{d}W_\epsilon/\mathrm{d}x$

tau_ss (real array) Steady-state mode torque $\tau_{\rm ss}$, in units of GM^2/R. Evaluated by integrating $\mathrm{d}\tau_{\rm ss}/\mathrm{d}x$

tau_tr (real array) Transient total mode torque $\tau_{\rm tr}$, in units of GM^2/R. Evaluated by integrating $\mathrm{d}\tau_{\rm tr}/\mathrm{d}x$

Rotation

beta (real array) Rotation splitting coefficient β. Evaluated by integrating $\mathrm{d}\beta/\mathrm{d}x$

Stellar Structure

M_star (real scalar) stellar mass, in units of g[1]

R_star (real scalar) stellar radius, in units of cm[1]

L_star (real scalar) stellar luminosity, in units of erg s^{-1}[1]

Delta_p (real array) Asymptotic p-mode large frequency separation $\Delta\nu$, in units of $\sqrt{GM/R^3}$

Delta_g (real array) Asymptotic g-mode inverse period separation $(\Delta P)^{-1}$, in units of $\sqrt{GM/R^3}$

[1] This option is only available when `model_type` is `'EVOL'`

1.9.2 Detail Files

A detail file gathers together information about a single mode found during a GYRE run. The data written to a detail file is controlled by the `detail_item_list` parameter of the `&ad_output` namelist group (for adiabatic calculations) and the `&nad_output` namelist group (for nonadiabatic calculations). This parameter is a comma-separated list of items to appear in the detail files. The items come in two flavors:

- *scalar* items comprise a single value, typically pertaining either to the star as a whole (i.e., a global quantity) or to a specific location in the star
- *array* items comprise a sequence of values, with each value pertaining to a single point in the discrete grid used to solve the oscillation equations. The sequence runs from the inner boundary to the outer boundary

The following subsections describe the items that may appear in a `detail_item_list` parameter, grouped together by functional area.

Solution Data

`n` (integer scalar) Number of grid points n

`x` (real array) Fractional radius $x \equiv r/R$

`y_1` (complex array) Solution variable y_1, defined in equations.pdf

`y_2` (complex array) Solution variable y_2, defined in equations.pdf

`y_3` (complex array) Solution variable y_3, defined in equations.pdf

`y_4` (complex array) Solution variable y_4, defined in equations.pdf

`y_5` (complex array) Solution variable y_5, defined in equations.pdf

`y_6` (complex array) Solution variable y_6, defined in equations.pdf

`omega` (complex scalar) Dimensionless eigenfrequency ω

Observables

`freq` (complex scalar) Dimensioned eigenfrequency. The units and reference frame are controlled by `freq_units` and `freq_frame` parameters of the `&ad_output` and `&nad_output` namelist groups

`freq_units` (character scalar) Units of `freq`

`freq_frame` (character scalar) Reference frame of `freq`

`f_T` (real scalar) Effective temperature perturbation amplitude $f_{\rm T}$. Evaluated using eqn. 5 of Dupret et al. (2003)

`f_g` (real scalar) Effective gravity perturbation amplitude $f_{\rm g}$. Evaluated using eqn. 6 of Dupret et al. (2003)

`psi_T` (real scalar) Effective temperature perturbation phase $\psi_{\rm T}$. Evaluated using eqn. 5 of Dupret et al. (2003)

`psi_g` (real scalar) Effective gravity perturbation phase $\psi_{\rm g}$

Classification & Validation

`j` (integer scalar) Unique mode index j. The first mode found during the GYRE run has $j = 1$, the second $j = 2$, and so on

`l` (integer scalar) Harmonic degree ℓ

`l_i` (complex scalar) Effective harmonic degree at inner boundary $\ell_{\rm i}$

m (integer scalar) Azimuthal order m

n_p (integer scalar) Acoustic-wave winding number $n_{\rm p}$

n_g (integer scalar) Gravity-wave winding number $n_{\rm g}$

n_pg (integer scalar) Radial order $n_{\rm pg}$ within the Eckart-Scuflaire-Osaki-Takata scheme (see Takata, 2006)

omega_int (complex scalar) Dimensionless eigenfrequency ω from integral expression. Evaluated using eqn. 1.71 of Marc-Antoine's Dupret's PhD thesis

Yt_1 (complex array) Primary eigenfunction for Takata classification $\mathcal{Y}_1$. Evaluated using a rescaled eqn. 69 of Takata (2006)

Yt_2 (complex array) Secondary eigenfunction for Takata classification $\mathcal{Y}_2$. Evaluated using a rescaled eqn. 70 of Takata (2006)

I_0 (complex array) First integral for radial modes I_0. Evaluated using eqn. 42 of Takata (2006)

I_1 (complex array) First integral for dipole modes I_1. Evaluated using eqn. 43 of Takata (2006)

prop_type (complex array) Propagation type ϖ based on local dispersion relation. $\varpi = 1$ in acoustic-wave regions, $\varpi = -1$ in gravity-wave regions, and $\varpi = 0$ in evanescent regions

Perturbations

x_ref (real scalar) Fractional radius of reference location $x_{\rm ref}$

xi_r_ref (complex scalar) Radial displacement perturbation $\xi_{\rm r}$ at reference location $x_{\rm ref}$, in units of R

xi_h_ref (complex scalar) Horizontal displacement perturbation $\xi_{\rm h}$ at reference location $x_{\rm ref}$, in units of R

eul_phi_ref (complex scalar) Eulerian potential perturbation Φ' at reference location $x_{\rm ref}$, in units of GM/R

deul_phi_ref (complex scalar) Eulerian potential gradient perturbation $\mathrm{d}\Phi'/\mathrm{d}x$ at reference location $x_{\rm ref}$, in units of GM/R^2

lag_S_ref (complex scalar) Lagrangian specific entropy perturbation δS at reference location $x_{\rm ref}$, in units of c_P

lag_L_ref (complex scalar) Lagrangian radiative luminosity perturbation $\delta L_{r,\rm R}$ at reference location $x_{\rm ref}$, in units of L

xi_r (complex array) Radial displacement perturbation $\xi_{\rm r}$, in units of R

xi_h (complex array) Horizontal displacement perturbation $\xi_{\rm h}$, in units of R

eul_phi (complex array) Eulerian potential perturbation Φ', in units of GM/R

deul_phi (complex array) Eulerian potential gradient perturbation $\mathrm{d}\Phi'/\mathrm{d}x$, in units of GM/R^2

lag_S (complex array) Lagrangian specific entropy perturbation δS, in units of c_P

lag_L (complex array) Lagrangian radiative luminosity peturbation $\delta L_{r,\rm R}$, in units of L

eul_P (complex array) Eulerian total pressure perturbation P', in units of P

eul_rho (complex array) Eulerian density perturbation ρ', in units of ρ

eul_T (complex array) Eulerian temperature perturbation T', in units of T

lag_P (complex array) Lagrangian total pressure perturbation δP, in units of P

lag_rho (complex array) Lagrangian density perturbation $\delta\rho$, in units of ρ

lag_T (complex array) Lagrangian temperature perturbation δT, in units of T

Energetics & Transport

eta (real scalar) Normalized growth rate η. Evaluated using expression in text of page 1186 of Stellingwerf (1978)

E [(real scalar)] Mode inertia E, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$

E_p (real scalar) Acoustic inertia E_p, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$ in acoustic-wave propagation regions

E_g (real scalar) Gravity inertia E_g, in units of MR^2. Evaluated by integrating $\mathrm{d}E/\mathrm{d}x$ in gravity-wave propagation regions

E_norm (real scalar) Normalized inertia E_norm. The normalization is controlled by the `inertia_norm` parameter of the `&osc` namelist group

E_ratio (real scalar) Ratio of mode inertia inside/outside the reference location x_ref

H (real scalar) Mode energy H, in units of GM^2/R

W (real scalar) Mode work W, in units of GM^2/R. Evaluated by integrating $\mathrm{d}W/\mathrm{d}x$

W_eps (real scalar) Mode nuclear work W_ϵ, in units of GM^2/R. Evaluated by integrating $\mathrm{d}W_\epsilon/\mathrm{d}x$

tau_ss (real scalar) Steady-state mode torque τ_ss, in units of GM^2/R. Evaluated by integrating $\mathrm{d}\tau_\mathrm{ss}/\mathrm{d}x$

tau_tr (real scalar) Transient total mode torque τ_tr, in units of GM^2/R. Evaluated by integrating $\mathrm{d}\tau_\mathrm{tr}/\mathrm{d}x$

dE_dx (real array) Differential inertia $\mathrm{d}E/\mathrm{d}x$, in units of MR^2

dW_dx (real array) Differential work $\mathrm{d}W/\mathrm{d}x$, in units of GM^2/R. Evaluated using eqn. 25.9 of Unno et al. (1989)

dW_eps_dx (real array) Differential nuclear work $\mathrm{d}W_{epsilon}/\mathrm{d}x$, in units of GM^2/R. Evaluated using eqn. 25.9 of Unno et al. (1989)

dtau_dx_ss (real array) Steady-state differential torque $\mathrm{d}\tau_\mathrm{ss}/\mathrm{d}x$, in units of GM^2/R

dtau_dx_tr (real array) Transient differential torque $\mathrm{d}\tau_\mathrm{tr}/\mathrm{d}x$, in units of GM^2/R

alpha_0 (real array) Excitation coefficient α_0. Evaluated using eqn. 26.10 of Unno et al. (1989)

alpha_1 (real array) Excitation coefficient α_1. Evaluated using eqn. 26.12 of Unno et al. (1989)

Rotation

beta (real scalar) Rotation splitting coefficient β. Evaluated by integrating $\mathrm{d}\beta/\mathrm{d}x$

dbeta_dx (real array) Unnormalized rotation splitting kernel $\mathrm{d}\beta/\mathrm{d}x$. Evaluated using eqn. 3.357 of Aerts et al. (2010)

lambda (complex array) Angular eigenvalue λ

Stellar Structure

M_star (real scalar) stellar mass, in units of g[1]

R_star (real scalar) stellar radius, in units of cm[1]

L_star (real scalar) stellar luminosity, in units of $\mathrm{erg\,s^{-1}}$[1]

Delta_p (real scalar) Asymptotic p-mode large frequency separation $\Delta\nu$, in units of $\sqrt{GM/R^3}$

Delta_g (real scalar) Asymptotic g-mode inverse period separation $(\Delta P)^{-1}$, in units of $\sqrt{GM/R^3}$

[1] This option is only available when `model_type` is `'EVOL'`

V_2 (real array) Dimensionless structure coefficient V_2, defined in equations.pdf

As (real array) Dimensionless structure coefficient A^*, defined in equations.pdf

U (real array) Dimensionless structure coefficient U, defined in equations.pdf

c_1 (real array) Dimensionless structure coefficient c_1, defined in equations.pdf

Gamma_1 (real array) Adiabatic exponent Γ_1, equations.pdf

nabla (real array) Dimensionless temperature gradient ∇, defined in equations.pdf

nabla_ad (real array) Adiabatic tempertature gradient $\nabla_{\rm ad}$, defined in equations.pdf

dnabla_ad (real array) Dimensionless gradient $\partial\nabla_{\rm ad}$, defined in equations.pdf

delta (real array) Thermodynamic coefficient δ, defined in equations.pdf

c_lum (real array) Dimensionless structure coefficient $c_{\rm lum}$, defined in equations.pdf

c_rad (real array) Dimensionless structure coefficient $c_{\rm rad}$, defined in equations.pdf

c_thn (real array) Dimensionless structure coefficient $c_{\rm thn}$, defined in equations.pdf

c_thk (real array) Dimensionless structure coefficient $c_{\rm thk}$, defined in equations.pdf

c_eps (real array) Dimensionless structure coefficient c_ϵ, defined in equations.pdf

eps_rho (real array) Energy generation partial ϵ_ρ, defined in equations.pdf

eps_T (real array) Energy generation partial ϵ_T, defined in equations.pdf

kap_rho (real array) Opacity partial κ_ρ, defined in equations.pdf

kap_T (real array) Opacity partial κ_T, defined in equations.pdf

Omega_rot (real array) Rotation angular frequency, in units of $\sqrt{GM/R^3}$

M_r (real array) Mass coordinate, in units of g^1

P (real array) Total pressure, in units of $\mathrm{dyn\,cm}^{-21}$

rho (real array) Density, in units of $\mathrm{g\,cm}^{-31}$

T (real array) Temperature, in units of K^1

1.9.3 File Formats

The format of sumamry and mode files depends on the value of the `summary_file_format` and `mode_file_format` parameters in the `&ad_output` and `&nad_output` namelist groups (see the *Output Parameters* section). Possible choices are:

- `'HDF'` : A binary format based on the HDF5 format
- `'TXT'` : A text format modeled after MESA's profile file format

Each has advantages and disadvantages. HDF files store floating-point data at full precision, and are very compact; however, they require additional tools to read them (e.g., the h5py Python package). TXT files, in contrast, are human-readable and therefore more accessible, but this comes at the price of larger files and the possible loss of precision in floating-point data.

For both formats, the data stored in the files come in two flavors – scalars (a single value) and arrays (a sequence of values)

HDF Format

GYRE's HDF-format output files adhere to the following conventions:

- All data objects are attached to the root HDF5 group (*/*)
- Attributes are used to store scalar data
- Datasets are used to store array data
- Real values are written with type *H5T_IEEE_F64LE* when GYRE is compiled in double precision (the default), and type *H5T_IEEE_F32LE* otherwise
- Integer values are written with type *H5T_STD_I32LE*
- Complex values are written as a compound type, composed of a real component *re* and an imaginary component *im*; the types of these components are the same as for real values

To simplify the process of reading HDF5 output produced by GYRE, the *gyre.py* Python module provides the *read_output* routine which can read both summary files and mode files (see the python_support chapter for more details).

TXT Format

GYRE's TXT-format files adhere to the following conventions:

- The first three lines contain the scalar data:
  - The first line contains the column numbers for the scalar data, starting at 1
  - The second line contains the column labels for the scalar data
  - The third line contains the actual scalar data values
- The subsequent lines contain the array data:
  - The fourth line contains the column numbers for the array data, starting at 1
  - The fifth line contains the the column labels for the array data
  - The sixth and subsequent lines contain the actual array data (one line per array element)
- Complex values are written as two columns, with the first column containing the real component and the second the imaginary component

1.9.4 Output Filters

1.10 Stellar Models

This chapter documents the different types of stellar models that can be used with GYRE to specify the equilibrium stellar configuration. Models fall into three categories, corresponding to the three possible choices for the `model_type` parameter of the `&model` namelist group (see the *Stellar Model Parameters* section):

1.10.1 Evolutionary Models

Evolutionary models are read from a file created by a separate stellar evolution code. The format of this file is specified by the `file_format` parameter of the `&model` namelist group (see the *Stellar Model Parameters* section). Possible choices are:

AMDL Binary file describing an evolutionary model in ASTEC format (as reverse engineered from the ADIPLS stellar oscillation code; see Christensen-Dalsgaard, 2008).

FAMDL Text file describing an evolutionary model in FAMDL format (as specified in the `CoRoT/ESTA File Formats` document).

FGONG Text file describing an evolutionary model in FGONG format (as specified in the updated `FGONG Format` document).

GSM HDF5 file describing an evolutionary model in GYRE Stellar Model (GSM) format.

MESA Text file describing an evolutionary model in MESA format (as specified in the *MESA Model Files* section).

B3 HDF5 file describing an evolutionary model in B3 format. This format is for testing purposes only, and will eventually be superseded and/or removed.

LOSC Text file describing an evolutionary model in the revised LOSC format.

OSC Text file describing an evolutionary model in OSC format (as specified in the `CoRoT/ESTA File Formats` document).

For all of these model formats, cubic spline interpolation is used to evaluate data between model grid points. The `deriv_type` parameter in the `&model` namelist group controls how the spline derivatives are set up.

1.10.2 Polytropic Models

Polytropic models represent stars in hydrostatic equilibrium that follow the polytropic equation of state,

To use a polytropic model with GYRE, set the `model_type` parameter in the :n ml_g:*model* namelist group to `'POLY'`, and the `file` parameter to the name of the polytropic structure file.

1.10.3 Homogeneous Models

Homogeneous models represent uniform-density stars in hydrostatic equilibrium; they are equivalent to polytropes with index $= 0$. Because their structure can be calculated analytically, GYRE creates them on-the-fly without the need to read from an external file.

To use a homogeneous model with GYRE, set the `model_type` parameter in the `&model` namelist group to `'HOM'`, and set the `Gamma_1`, `n`, `s` and `grid_type` parameters to suitable values (see the *Stellar Model Parameters* section for more details).

1.10.4 MESA Model Files

MESA-format files store data describing a MESA stellar model in an ASCII text file. (Note that MESA itself refers to these files as 'GYRE-format' files. To create one of these files, set the `pulse_data_format` parameter of the `&controls` namelist group to the value `GYRE`).

There are a number of versions of the MESA format, distinguished by the initial header line:

Version 0.01

The first line of version-0.01 MESA files is a header with the following columns:

Column	Variable	Datatype	Definition
1	n	integer	number of grid points
2	M	real	stellar mass ()
3	R	real	photospheric radius ()
4	L	real	photospheric luminosity ($^{-1}$)

The subsequent n lines contain the model data, one line per grid point extending from the center to the surface, with the following columns:

Column	Variable	Datatype	Definition
1	k	integer	grid point index ($k = 1, \ldots, n$)
2	r	real	radial coordinate ()
3	w	real	transformed mass coordinate $M_r/(M - M_r)$
4	L_r	real	luminosity ($^{-1}$)
5	p	real	total pressure ($^{-2}$)
6	T	real	temperature ()
7	ρ	real	density ($^{-3}$)
8	∇	real	dimensionless temperature gradient $\ln T \ln p$
9	N^2	real	Brunt-Väisälä frequency squared ($^{-2}$)
10	c_v	real	specific heat at constant volume ($^{-1}$ $^{-1}$)
11	c_p	real	specific heat at constant pressure ($^{-1}$ $^{-1}$)
12	χ_T	real	equation-of-state partial $(\ln p \ln T)_\rho$
13	χ_ρ	real	equation-of-state partial $(\ln p \ln \rho)_T$
14	κ	real	opacity (^{2} $^{-1}$)
15		real	opacity partial $(\ln \kappa \ln T)_\rho$
16		real	opacity partial $(\ln \kappa \ln \rho)_T$
17	ϵ	real	total energy generation rate (s^{-1} $^{-1}$)
18		real	nuclear energy generation rate partial $(\ln \ln T)_\rho$ (s^{-1} $^{-1}$)
19		real	nuclear energy generation rate partial $(\ln \ln \rho)_T$ (s^{-1} $^{-1}$)

Version 0.19

The first line of version-0.19 MESA files is a header with the following columns:

Column	Variable	Datatype	Definition
1	n	integer	number of grid points
2	M	real	stellar mass ()
3	R	real	photospheric radius ()
4	L	real	photospheric luminosity ($^{-1}$)
5	19	integer	version number $\times 100$

The subsequent n lines contain the model data, one line per grid point extending from the center to the surface, with the following columns:

Column	Variable	Datatype	Definition
1	k	integer	grid point index ($k = 1, \ldots, n$)
2	r	real	radial coordinate ()
3	w	real	transformed mass coordinate $M_r/(M - M_r)$
4	L_r	real	luminosity ($^{-1}$)
5	p	real	total pressure ($^{-2}$)
6	T	real	temperature ()
7	ρ	real	density ($^{-3}$)
8	∇	real	dimensionless temperature gradient $\ln T \ln p$
9	N^2	real	Brunt-Väisälä frequency squared ($^{-2}$)
10	Γ_1	real	first adiabatic exponent $(\ln p \ln \rho)_{\rm ad}$
11	$\nabla_{\rm ad}$	real	adiabatic temperature gradient $(\ln p \ln T)_{\rm ad}$
12	δ	real	dimensionless thermal expansion coefficient $-(\ln \rho \ln T)_p$
13	κ	real	opacity (^{2} $^{-1}$)
14		real	opacity partial $(\ln \kappa \ln T)_\rho$
15		real	opacity partial $(\ln \kappa \ln \rho)_T$
16	ϵ	real	total energy generation rate (s^{-1} $^{-1}$)
17		real	nuclear energy generation rate partial $(\ln \ln T)_\rho$ (s^{-1} $^{-1}$)
18		real	nuclear energy generation rate partial $(\ln \ln \rho)_T$ (s^{-1} $^{-1}$)
19	$\Omega_{\rm rot}$	real	rotation angular velocity ($^{-1}$)

Version 1.00

The first line of version-1.00 MESA files is a header with the following columns:

Column	Variable	Datatype	Definition
1	n	integer	number of grid points
2	M	real	stellar mass ()
3	R	real	photospheric radius ()
4	L	real	photospheric luminosity ($^{-1}$)
5	100	integer	version number $\times 100$

The subsequent n lines contain the model data, one line per grid point extending from the center to the surface, with the following columns:

Column	Variable	Datatype	Definition
1	k	integer	grid point index ($k = 1, \ldots, n$)
2	r	real	radial coordinate ()
3	w	real	transformed mass coordinate $M_r/(M - M_r)$
4	L_r	real	luminosity ($^{-1}$)
5	p	real	total pressure ($^{-2}$)
6	T	real	temperature ()
7	ρ	real	density ($^{-3}$)
8	∇	real	dimensionless temperature gradient $\ln T \ln p$
9	N^2	real	Brunt-Väisälä frequency squared ($^{-2}$)
10	Γ_1	real	first adiabatic exponent $(\ln p \ln \rho)_{\rm ad}$
11	$\nabla_{\rm ad}$	real	adiabatic temperature gradient $(\ln p \ln T)_{\rm ad}$
12	δ	real	dimensionless thermal expansion coefficient $-(\ln \rho \ln T)_p$
13	κ	real	opacity (^{2} $^{-1}$)
14	κ	real	opacity partial $\kappa(\ln \kappa \ln T)_\rho$ (^{2} $^{-1}$)
15	κ	real	opacity partial $\kappa(\ln \kappa \ln \rho)_T$ (^{2} $^{-1}$)
16	ϵ	real	total energy generation rate (s^{-1} $^{-1}$)
17		real	nuclear energy generation rate partial $(\ln \ln T)_\rho$ (s^{-1} $^{-1}$)
18		real	nuclear energy generation rate partial $(\ln \ln \rho)_T$ (s^{-1} $^{-1}$)
19	$\Omega_{\rm rot}$	real	rotation angular velocity ($^{-1}$)

Version 1.01

The first line of version-1.01 MESA files is a header with the following columns:

Column	Variable	Datatype	Definition
1	n	integer	number of grid points
2	M	real	stellar mass ()
3	R	real	photospheric radius ()
4	L	real	photospheric luminosity ($^{-1}$)
5	101	integer	version number $\times 100$

The subsequent n lines contain the model data, one line per grid point extending from the center to the surface, with the following columns:

Column	Variable	Datatype	Definition
1	k	integer	grid point index ($k = 1, \ldots, n$)
2	r	real	radial coordinate ()
3	w	real	transformed mass coordinate $M_r/(M - M_r)$
4	L_r	real	luminosity ($^{-1}$)
5	p	real	total pressure ($^{-2}$)
6	T	real	temperature ()
7	ρ	real	density ($^{-3}$)
8	∇	real	dimensionless temperature gradient $\ln T \ln p$
9	N^2	real	Brunt-Väisälä frequency squared ($^{-2}$)
10	Γ_1	real	first adiabatic exponent $(\ln p \ln \rho)_{\rm ad}$
11	$\nabla_{\rm ad}$	real	adiabatic temperature gradient $(\ln p \ln T)_{\rm ad}$
12	δ	real	dimensionless thermal expansion coefficient $-(\ln \rho \ln T)_p$
13	κ	real	opacity (^{2} $^{-1}$)
14	κ	real	opacity partial $\kappa (\ln \kappa \ln T)_\rho$ (^{2} $^{-1}$)
15	κ	real	opacity partial $\kappa (\ln \kappa \ln \rho)_T$ (^{2} $^{-1}$)
16		real	nuclear energy generation rate (s^{-1} $^{-1}$)
17		real	nuclear energy generation rate partial $(\ln \ln T)_\rho$ (s^{-1} $^{-1}$)
18		real	nuclear energy generation rate partial $(\ln \ln \rho)_T$ (s^{-1} $^{-1}$)
19	$\Omega_{\rm rot}$	real	rotation angular velocity ($^{-1}$)

1.11 Mathmatical Formalism

This chapter details the mathematical formalism on which GYRE is built.

1.11.1 Physical Formulation

Fluid Equations

The basis for most subsequent equations are the fluid equations, comprising the conservation laws for mass

$$\rho t + \cdot \nabla (\rho) = 0$$

and momentum

$$\rho (t + \cdot \nabla) = -\nabla P - \rho \nabla \Phi;$$

the heat equation

$$\rho T (t + \cdot \nabla) S = \rho - \nabla \cdot;$$

and Poisson's equation

$$\nabla^2 \Phi = 4\pi G \rho.$$

Here, ρ, p, T, S and are the fluid density, pressure, temperature, specific entropy and velocity; while Φ is the gravitational potential, is the specific nuclear energy generation rate and the energy flux.

The energy flux is the sum of the radiative () and convective () fluxes,

$$= +;$$

the radiative flux is given by the radiative diffusion equation,

$$= \frac{c}{3\kappa\rho}\nabla(aT^4),$$

where κ is the opacity and a the radiation constant.

Thermodynamic Relations

The fluid equations are augmented by the thermodynamic relationships between the four state variables (p, T, ρ and S). Only two of these are required to uniquely specify the state (we assume that the composition remains fixed over an oscillation cycle). In GYRE, p and S are adopted as these primary variables, and the other two are presumed to be derivable from them:

$$\rho = \rho(p, S), \qquad T = T(p, s).$$

Equilibrium State

In a static equilibrium state the fluid velocity vanishes. The momentum equation then becomes the hydrostatic equilibrium equation

$$\nabla P = -\rho\nabla\Phi.$$

Assume the equilibrium state is spherically symmetric, this simplifies to

$$pr = -\rho\Phi r.$$

Poisson's equation can be integrated once to yield

$$\Phi r = \frac{G}{r^2}\int 4\pi\rho r^2\, r = \frac{GM_r}{r^2},$$

and so the hydrostatic equilibrium equation becomes

$$pr = -\rho\frac{GM_r}{r^2}.$$

The heat equation in the equilibrium state is

$$\rho TSt = \rho - \nabla\cdot.$$

If the star is in thermal equilibrium then the left-hand side vanishes, and the nuclear heating rate balances the flux divergence term.

Linearized Equations

Applying an Eulerian (fixed position, denoted by a prime) perturbation to the mass and momentum conservation equations, they linearize about the static equilibrium state as

$$\rho' + \nabla\cdot(\rho') = 0,$$

$$\rho' t = -\nabla P' + \frac{\rho'}{\rho}\nabla P - \rho\nabla\Phi',$$

$$\nabla^2\Phi' = 4\pi G\rho'$$

Likewise applying a Lagrangian (fixed mass element, denoted by a δ) perturbation to the heat equation and the thermodynamic relations, they linearizes about the equilibrium state as

$$T\delta St = \delta - \delta\left(\frac{1}{\rho}\nabla\cdot\right).$$

$\Gamma_1 \frac{\delta p}{p} - \upsilon_T \frac{\delta S}{c_p}$

$$\frac{\delta T}{T} = \nabla_{\rm ad}\frac{\delta p}{p} + \frac{\delta S}{c_p}$$

The absence of either a prime or a δ denotes an equilibrium quantity. No $'$ terms appear because the equilibrium state is static. The thermodynamic partial derivatives are defined as

$$\Gamma_1 = (\ln p \ln \rho)_S \qquad \upsilon_T = (\ln \rho \ln T)_p \qquad c_p = (S \ln T_p \qquad \nabla_{\rm ad} = (Tp)_S$$

Separation

With a separation of variables in spherical-polar coordinates (r, θ, ϕ), and assuming an oscillatory time (t) dependence with angular frequency σ, solutions to the linearized fluid equation can be expressed as

$$(r, \theta, \phi; t) = \mathrm{Re}\left[\sqrt{4\pi}\,(r)\,Y_\ell^m(\theta,\phi)\,\exp(-\sigma t)\right],$$

$$(r, \theta, \phi; t) = \mathrm{Re}\left[\sqrt{4\pi}\,(r)\,rY_\ell^m(\theta,\phi)\,\exp(-\sigma t)\right],$$

$$f'(r, \theta, \phi; t) = \mathrm{Re}\left[\sqrt{4\pi}\,'(r)\,Y_\ell^m(\theta,\phi)\,\exp(-\sigma t)\right]$$

Here, is the radial component of the displacement perturbation vector , and is the corresponding horizontal (polar and azimuthal) part of this vector; is the horizontal part of the spherical-polar gradient operator; Y_ℓ^m is the spherical harmonic with harmonic degree ℓ and azimuthal order m; and f stands for any perturbable scalar. The displacement perturbation vector is related to the velocity perturbation via

$$' = t$$

1.11.2 Oscillation Equations

The oscillation equations follow from substituting the above solution forms into the linearized equations:

$$' + \frac{1}{r^2}r\left(r^2\right) - \frac{\ell(\ell+1)}{r}\rho = 0,$$

1.12 Troubleshooting

1.13 Building Polytropic Structure Files

This appendix describes the **`build_poly`** executable, which creates polytropic structure files for use with GYRE.

1.13.1 Pre-Requisites

In addition to GYRE's general pre-requisites (see the *Installation* chapter), **`build_poly`** needs a thread-safe version of the ODEPACK ordinary differential integrator library. This library is shipped with version 20.3.2 (and more recent) of the MESA SDK.

On Linux and MacOS platforms, this library is bundled together with all of the other pre-requisites in the Madison Software Development Kit (SDK), which can be downloaded from the MESA SDK homepage. Using this SDK is strongly recommended.

It does this by solving the Lane-Emden equation

$$\frac{1}{\xi^2}\xi\left(\xi^2\Theta\xi\right) = -\Theta$$

for constant polytropic index . Here, the dependent variable Θ represents the local density ρ via

Compile

Index