Notes on adiabatic oscillation programme

Fourth edition

October 15, 1997

Jørgen Christensen-Dalsgaard
Institut for Fysik og Astronomi, Aarhus Universitet

Notes on revisions: These notes were originally written at HAO in 1983. The present edition was produced in September 1997, following a major reorganization of the code involving also the addition or change of several input parameters. It is believed to be consistent with the versions of the programme implemented on the Convex C38 in Aarhus, and the PowerChallenges in Stanford (`fault.stanford.edu`), at JILA (`hydra.colorado.edu`) and Aarhus (and `bigcat.obs.aau.dk`) at that time. This version is more recent than versions that have been distributed over the past few years, including the first GONG release (September 1 1994).

It is recommended to consider also the notes given in the source to subroutine `adipls`, particularly as far as input parameters to the programme are concerned, since the notes in the programme source are generally updated when the programme is changed.

Contents

1 Introduction

These notes describes a programme for calculation of adiabatic oscillations of stellar models. This has been developed over several years, with a number of different uses in mind, and is therefore fairly complex both in structure and in the control parameters possible. In particular it has not been designed for general use, and so it has not been thoroughly tested in all conceivable cases. On the other hand the programme is quite flexible, both in the physical situations it may consider, and in the numerical schemes it employs. Thus both models with vanishing surface pressure (and hence a singularity at the surface), and physical models terminating at a finite pressure can be considered; and it is possible to consider models truncated at a finite distance from the centre, as well as complete models.

A principal goal of the present notes has been to give a precise definition of the calculations carried out by the programme. The notes also provide information intended to be sufficiently detailed to permit easy use of the programme, describing the notation used, the form needed for the equilibrium model that must be provided to the programme, and the output produced, as well as some notes on the experience gained so far in running the programme. However, it is sufficiently complex that modifications by users other than the author should be attempted only with care.

The programme uses equilibrium models in the form of so-called `amdl` files, produced by the JC-D evolution code. They consist of an essentially minimal set of variables for computing adiabatic oscillations. The variables are described in Section 5, and the input format is provided in Section 7.1. The programme is controlled by a vast number of control parameters, described in Section 7.2. In most cases, particularly for computing p-mode frequencies of solar models, these parameters barely need to be changed. Samples of suitable input files can be provided.

The output from the programme is described in Section 8. The most important output is the set of frequencies and other mode data, and the mode eigenfunctions. The most extensive set of mode data is provided in binary form in the *grand summary file* (Section 8.2), which also gives diagnostics on the computation. A more compact set of mode data, also in binary form, is given in the *short summary* (Section 8.3). The eigenfunctions can be provided as a comprehensive set, a set containing just the displacement amplitudes, or a set giving the density-weighted displacement amplitudes (Section 8.4); the latter set is the most appropriate for setting up kernels. The eigenfunction data contain the grand summary. Note that, as the eigenfunction files are rather large, it is advisable to output them only if explicitly needed. The programme may also output kernels for frequency spitting caused by spherically symmetric rotation (Section 8.5) and, less usefully, kernels for the effect on frequencies of changes in the adiabatic exponent Γ_1 at fixed density ρ (Section 8.6).

In case of problems, diagnostic output is provided in a log file with the default name `adipls-status.log` (see Section 8.8; the name may be changed in the input file). The programme also produces output summarizing the calculation, and providing further diagnostics. This is typically less useful, except for debugging; in particular, reports of problems should be accompanied by this output file (see Section 8.1).

In Section 2 the equations and boundary conditions are briefly discussed; this section also serves to define the notation used. A more detailed description of the equations is contained in Appendix A. Section 3 describes the numerical techniques used to solve the equations. Section 4 defines the quantities computed by the programme. Section 6 provides a brief discussion of the notation and data storage used in the programme, mainly serving to define the terms used in Section 7. Finally Section 9 describes the main programme that must be supplied by the user to set up storage for the calculation.

Very few references are given here. This evidently does not imply that the methods discussed are original. A detailed discussion of non-radial stellar oscillations, as well as numerous references, may be found, *e.g.*, in Unno *et al.* (1989).

I finally note that a separate short set of notes are available on the use of this and other related programmes: *Notes on using the solar models and adiabatic pulsations package.*

2 Notation. Equations and boundary conditions

The notation used here in general follows that of Christensen-Dalsgaard (1981, 1982; in the following CD81 and CD82); however, it is described in part here. Words entirely in `typewriter type` refer to names of variables in the programme.

2.1 Variables and equations

As usual the perturbations are assumed to be separated in θ and ϕ in terms of spherical harmonics, and the time dependence to be given as a harmonic function. Thus the displacement vector is written as

$$\delta\vec{r} = \mathrm{Re}\left\{\left[\xi_r(r)Y_l^m(\theta,\phi)\mathbf{a}_r + \xi_h(r)\left(\frac{\partial Y_l^m}{\partial\theta}\mathbf{a}_\theta + \frac{1}{\sin\theta}\frac{\partial Y_l^m}{\partial\phi}\mathbf{a}_\phi\right)\right]\exp(-i\omega t)\right\} . \qquad (2.1)$$

Similarly the perturbation in, e.g. pressure, may be written

$$\delta p = \mathrm{Re}\left[\delta p(r)Y_l^m(\theta,\phi)\exp(-i\omega t)\right] . \qquad (2.2)$$

Here Y_l^m is a spherical harmonic, $\mathbf{a}_r$, $\mathbf{a}_\theta$, and $\mathbf{a}_\phi$ are unit vectors in the r, θ, and ϕ directions and ω is the frequency. As the oscillations are adiabatic (and only conservative boundary conditions are considered) ω is real, and the amplitude functions $\xi_r(r)$, $\xi_h(r)$, $\delta p(r)$, etc. can be chosen to be real.

The equations for non-radial oscillations can be found in a number of references (*e.g.* Unno *et al.* 1989). However, the programme permits a modification of Poisson's equation for the perturbations, which is written as

$$\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}\Phi'}{\mathrm{d}r}\right) - \frac{l(l+1)}{r^2}\Phi' = -4\pi\lambda G\rho' \qquad (2.3)$$

(the ordinary equation clearly corresponds to having $\lambda = 1$). This was done to investigate the effect of making a gradual transition between the Cowling approximation, where Φ' is neglected, to the full set of equations (Christensen-Dalsgaard & Gough, in preparation); this can clearly be accomplished by varying λ continuously from 0 to 1.

Nonradial oscillations may be considered both in the Cowling approximation (corresponding to a second-order system of equations) and in the full case (corresponding to a fourth-order set). For radial oscillations the perturbation in the gravitational potential can be eliminated analytically, so that here the basic, complete set of equations is of second order.

A further restriction is that nonradial oscillations of a truncated model can only be treated in the Cowling approximation; this is almost inevitable, as there seems to be no natural way to specify the boundary conditions for the perturbation in the gravitational potential at the base of a truncated model.

In some equilibrium models a contribution to the equation of hydrostatic support from "turbulent pressure" may be included. This is the case, *e.g.*, for semi-empirical models of the solar atmosphere or models based on averages of hydrodynamical simulations. This raises the question of the appropriate treatment of the turbulent pressure in the perturbation equations. The programme allows various options for treating (or, more precisely, neglecting) effects of turbulent pressure. These are controlled by the definition of the equilibrium-model variables and/or the parameter `iturpr`. The details are discussed in Section A.3. It should be noted that effects of turbulent pressure may well dominate the differences between the observed solar frequencies and frequencies of solar models where turbulent pressure is neglected (*e.g.* Rosenthal *et al.* 1997).

The degree l is treated as a real variable (clearly only integer values have physical meaning). The transition between the nonradial and the radial equations is defined in the programme to take place at $l = 10^{-6}$.

In the calculation the frequency is expressed in terms of the dimensionless squared frequency

$$\sigma^2 = \frac{R^3}{GM}\omega^2 , \tag{2.4}$$

where M and R are mass and photospheric radius of the equilibrium model. The eigenfunctions are defined in terms of a set of dimensionless functions $y_i(x)$ where $x = r/R$. For nonradial oscillations ($l > 0$)

$$\begin{aligned} y_1 &= \frac{\xi_r}{R} , \\ y_2 &= x\left(\frac{p'}{\rho} - \Phi'\right)\frac{l(l+1)}{\omega^2 r^2} = \frac{l(l+1)}{R}\xi_h , \\ y_3 &= x\frac{\Phi'}{gr} , \\ y_4 &= x^2\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{y_3}{x}\right) . \end{aligned} \tag{2.5a}$$

For radial oscillations only y_1 and y_2 are used, where y_1 is defined as above, and

$$y_2 = \frac{p'}{\omega^2 R^2 \rho} . \tag{2.5b}$$

2.2 Removing the x^l variation near the centre

The dependent variables y_i have been chosen in such a way that for $l > 0$ they all vary as x^{l-1} for $x \to 0$. For large l a considerable (and fundamentally unnecessary) computational effort would be needed to represent this variation sufficiently accurately with, *e.g.*, a finite difference technique, if these variables were to be used in the numerical integration. Instead we introduce a new set of dependent variables by

$$\hat{y}_i = x^{-l+1} y_i, \qquad i = 1, 2, 3, 4 . \tag{2.6}$$

These variables are then $O(1)$ in x near the centre. They are used in the region where the variation in the y_i is dominated by the x^{l-1} behaviour. Specifically one obtains from JWKB theory that

$$y_1(x) \sim \exp\left(\int^x k \mathrm{d}x\right) , \tag{2.7}$$

where

$$k^2 = \frac{l(l+1)}{x^2}\left(1 - \frac{\tilde{N}^2}{\sigma^2}\right)\left(1 - \frac{\sigma^2}{\tilde{S}_l^2}\right) , \tag{2.8}$$

and $\tilde{N}$ and $\tilde{S}_l$ are the dimensionless buoyancy and Lamb frequencies (see Section 5 below). Here $\tilde{N}^2$ and $\tilde{S}_l^{-2} \to 0$ as $x \to 0$. Thus close to the centre $k^2 \simeq l(l+1)/x^2$, and, to this order, the JWKB result is that $y_1 \sim x^{l+1/2}$. Writing $k^2 = h(x)l(l+1)/x^2$ this behaviour is clearly dominant as long as $h(x) > 1/4$. Thus the transition from $\hat{y}_i$ to y_i at a value $x_{\rm ev}$ of x where $h(x) = 1/4$, such that the exponentially growing solution dominates for $x > x_{\rm ev}$. The default is to choose the smallest value of x where $h(x) = 1/4$; however, there is the option of restricting the search to be outside a given value of x.

This transformation is applied when non-radial oscillations are computed. It permits calculating modes of arbitrary high degree in a complete model; when the $\hat{y}_i$ are multiplied by x^{l-1} to obtain the y_i this is done in a special routine that replaces x^{l-1} by zero when it is below the (machine-dependent) limit for underflow.

2.3 Inner boundary conditions

The inner conditions depend on whether one considers a full model, including the centre, or an envelope model. In the former case, the boundary conditions are regularity conditions (*e.g.* Christensen-Dalsgaard, Dilke & Gough 1974). In the latter case, there is considerably more flexibility in the choice of condition.

2.3.1 Full model

When the centre is included in the equilibrium model, the solution must satisfy regularity conditions at the innermost meshpoint. These are obtained by expansion around the centre, to terms that are $O(x^2)$ times the leading terms. It is useful to note that it follows from the expansion that

$$\xi_r \simeq l\xi_h \qquad \text{or} \qquad y_2 \simeq (l+1)y_1 \quad \text{for} \quad x \to 0 \; , \tag{2.9}$$

and

$$\frac{\mathrm{d}\Phi'}{\mathrm{d}r} \simeq \frac{l}{r}\Phi' \qquad \text{or} \qquad y_4 \simeq (l-2)y_3 \quad \text{for} \quad x \to 0 \; . \tag{2.10}$$

2.3.2 Truncated model

In a model truncated at a finite distance from the centre it is assumed that the calculation is for radial oscillations, or in the Cowling approximation. Here the programme allows four types of conditions, flagged by the input parameter `ibotbc` (*cf.* Section 7.2.3 below):

i) (`ibotbc = 0`) Applicable when the innermost meshpoint x_1 is in an evanescent region. The condition sets the relation between y_1 and y_2 so as to select the solution that decreases exponentially towards the interior. From the JWKB relation for y_1, equations (2.7) and (2.8), and the differential equation for y_1 one finds that approximately

$$y_2(x_1) \simeq \frac{x_1|k(x_1)|}{1-\sigma^2/\tilde{S}_l^2(x_1)} y_1(x_1) \; . \tag{2.11}$$

This condition is mainly useful for modes of relatively large degree.

ii) (`ibotbc = 1`) Specification of the ratio between y_1 and y_2 at the innermost mesh point x_1 by

$$y_1(x_1) = \alpha(1-F_{\mathrm{BC}}), \qquad y_2(x_1) = \alpha F_{\mathrm{BC}} \; , \tag{2.12}$$

where α is an arbitrary scale factor. Note that this condition can also be written as

$$F_{\mathrm{BC}} y_1(x_1) - (1-F_{\mathrm{BC}}) y_2(x_1) = 0 \; . \tag{2.12a}$$

Here F_{BC} is determined by the input parameter `fcttbc` (*cf.* Section 7.2.3).

iii) (`ibotbc = 2`) Setting

$$\frac{\mathrm{d}y_1}{\mathrm{d}x} = 0 \qquad \text{at} \quad x = x_1 \; . \tag{2.13}$$

This can be expressed as a relation between $y_1(x_1)$ and $y_2(x_1)$ by using the differential equation for y_1 (*cf.* Appendix A).

iv) (`ibotbc = 3`) Setting

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{y_1}{x}\right) = 0 \qquad \text{at} \quad x = x_1 \; . \tag{2.14}$$

This can be expressed as a relation between $y_1(x_1)$ and $y_2(x_1)$ by using the differential equation for y_1 (*cf.* Appendix A).

2.4 Surface conditions

The surface conditions depend on whether or not the surface pressure of the model vanishes. For vanishing surface pressure (as occurs, for example, in complete polytropic models) the surface is a singular point where regularity conditions must be imposed. When the pressure is non-vanishing, as in realistic stellar models truncated at a suitable point in the atmosphere, various conditions can be used. These are selected by the input parameters `istsbc` and `fctsbc`.

For nonradial oscillations the condition on the gravitational potential perturbation is common to the singular and nonsingular cases. It is obtained by demanding continuity of Φ' and its first derivative; in terms of the variables used here this may be expressed as

$$y_4(x_\mathrm{s}) = -[l + U(x_\mathrm{s})]y_3 + \lambda U(x_\mathrm{s})y_1 \; , \tag{2.15}$$

where x_s is the value of x at the surface, and $U = 4\pi\rho r^3/m$. For vanishing surface density, as would normally be the case for a singular surface, the terms in $U(x_\mathrm{s})$ vanish. However, for the iso-pycnic model (corresponding to constant density, *i.e.*, to a polytrope of index 0), $U(x_\mathrm{s}) = 3$.

2.4.1 Singular surface

When the surface pressure vanishes, the solution in the vicinity of the singular point is obtained by expansion around this singularity, retaining terms that are $O(t)$ times the leading order term, where t is the depth below the surface (it might be noted here that the expansion depends qualitatively on whether or not the surface density vanishes as well, *i.e.*, whether or not the effective polytropic index μ at the surface is non-zero). Details of this procedure, including expressions for the expansion coefficients, were given by Christensen-Dalsgaard & Mullan (1994).

2.4.2 Simple surface conditions

When the surface pressure is non-vanishing, as is the case for a realistic stellar model, the programme allows various simple expressions for the surface pressure condition. These are selected by setting `istsbc = 0`. One is to use the condition $\delta p = 0$, which in terms of the variables used here becomes

$$y_2(x_\mathrm{s}) = \frac{l(l+1)}{\sigma^2}[y_1(x_\mathrm{s}) - y_3(x_\mathrm{s})] \tag{2.16a}$$

in the nonradial case, or

$$y_2(x_\mathrm{s}) = \frac{1}{\sigma^2} y_1(x_\mathrm{s}) \tag{2.16b}$$

in the radial case.

To permit the use of different surface boundary conditions, a boundary condition on the form

$$y_2(x_\mathrm{s}) = \frac{l(l+1)}{\sigma^2}[(1 - F_\mathrm{SBC})y_1(x_\mathrm{s}) - y_3(x_\mathrm{s})] \tag{2.16c}$$

in the nonradial case, or

$$y_2(x_s) = \frac{1}{\sigma^2}(1 - F_{\rm SBC}) y_1(x_s) \qquad (2.16d)$$

in the radial case, may be used. Here $F_{\rm SBC}$ is an input parameter to the programme. For $F_{\rm SBC} = 0$ we recover the conditions (2.16a) or (2.16b), *i.e.*, $\delta p = 0$, whereas for $F_{\rm SBC} = 1$ the condition is equivalent to $p' = 0$, *i.e.*, the vanishing of the Eulerian pressure perturbation. $F_{\rm SBC}$ is determined by the input parameter `fctsbc`.

2.4.3 Match to the solution for an isothermal atmosphere

A more realistic condition is obtained by matching the solution onto the energetically confined of the two (analytically known) solutions for adiabatic waves in an isothermal atmosphere matched to the model (*cf.* Unno *et al.* 1989, pp 163 *ff*). This condition is selected by setting `istsbc = 1`. It has been implemented in the following form: Introduce

$$V_g = \frac{Gm\rho}{\Gamma_1 p r} , \qquad A_{\rm i} = V_g(\Gamma_1 - 1) , \qquad (2.17)$$

and set

$$\gamma = (A_{\rm i} + 4 - V_g)^2 + 4(\sigma^2 x^3 - A_{\rm i}) \left(\frac{l(l+1)}{\sigma^2 x^3} - V_g \right) , \qquad (2.18)$$

and

$$C = \frac{\frac{1}{2}(\gamma^{1/2} + V_g - A_{\rm i}) - 2}{V_g - \dfrac{l(l+1)}{\sigma^2 x^3}} , \qquad (2.19)$$

evaluated at $x = x_s$; then the condition in the nonradial case is

$$y_2(x_s) = \frac{l(l+1)}{\sigma^2 x_s^2} \left\{ C y_1(x_s) - \left[1 + \left(\frac{l(l+1)}{\sigma^2 x_s^3} - l - 1 \right) (V_g + A_{\rm i})^{-1} \right] y_3(x_s) \right\} , \qquad (2.20a)$$

and in the radial case

$$y_2(x_s) = \frac{C}{x^2 \sigma^2} y_1(x_s) . \qquad (2.20b)$$

Note that in the nonradial case the term in y_3 has been approximated by assuming that both the frequency and the degree is "small" (*cf.* Unno *et al.* 1989); if this is not the case y_3 is small and the term is almost certainly negligible (it may well have negligible influence on the solution in any case).

It should be noted that the expression for $A_{\rm i}$ is the value of A (*cf.* equation 5.1) in the isothermal case, where $\mathrm{d}\ln\rho/\mathrm{d}\ln r = \mathrm{d}\ln p/\mathrm{d}\ln r$. This is the form specifically used in the programme.

To use this condition we must clearly require that $\gamma \geq 0$; this approximately corresponds to requiring that the frequency be below the local acoustical cut-off frequency at $x = x_s$. If this were not the case, the solution in the isothermal atmosphere would have a running-wave component, and the system would no be longer conservative; thus the eigenfrequency and the eigenfunctions become complex, and the problem can no longer be treated with the present programme. If a solution is attempted above this limit, the condition corresponding to `istsbc = 0` is used, and a warning message is printed. Furthermore, the value of `istsbc` used in setting `icase` (*cf.* Section 8.2) is set to zero.

3 Numerical techniques

The numerical problem can be formulated generally as that of solving

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = \sum_{j=1}^{I} a_{ij}(x) y_j(x) \; , \qquad \text{for } i = 1, \ldots, I \; , \tag{3.1}$$

with the boundary conditions

$$\sum_{j=1}^{I} b_{ij} y_j(x_1) = 0 \; , \qquad \text{for } i = 1, \ldots , I/2 \; , \tag{3.2}$$

$$\sum_{j=1}^{I} c_{ij} y_j(x_\mathrm{s}) = 0 \; , \qquad \text{for } i = 1 \; , \ldots, I/2 \; . \tag{3.3}$$

Here the order I of the system is 4 for the full nonradial case, and 2 for radial oscillations or nonradial oscillations in the Cowling approximation. This system only allows non-trivial solutions for selected values of σ^2 which is thus an eigenvalue of the problem.

The programme permits solving these equations with two basically different techniques, each with some variants. The first is a shooting method, where solutions satisfying the boundary conditions are integrated separately from the inner and outer boundary, and the eigenvalue is found by matching these solutions at a suitable inner fitting point x_f (defined by the input parameter `xfit`). The second technique is to solve the equations, together with a normalization condition and either all, or all but one, of the boundary conditions using a relaxation technique; the eigenvalue is then found by requiring continuity of one of the eigenfunctions at an interior matching point when all the boundary conditions are satisfied, or by requiring that the remaining boundary condition be satisfied. The choice of integration method is controlled by the input parameter `mdintg` (*cf.* Section 7.2.3).

For simplicity we do not distinguish between $\hat{y}_i$ and y_i (*cf.* Section 2.2) in the bulk of this Section. It is implicitly understood that the dependent variable (which is denoted y_i) is $\hat{y}_i$ for $x < x_\mathrm{ev}$ and y_i for $x \geq x_\mathrm{ev}$. The numerical treatment of the transition between $\hat{y}_i$ and y_i is discussed in Section 3.3.

3.1 The shooting method

It is convenient here to distinguish between $I = 2$ and $I = 4$. For $I = 2$ the differential equations (3.1) have a unique (apart from normalization) solution $y_i^{(\mathrm{i})}$ satisfying the inner boundary conditions (3.2), and a unique solution $y_i^{(\mathrm{o})}$ satisfying the outer boundary conditions (3.3). These may be obtained by numerical integration of the equations. The final solution can then be represented as $y_j = C^{(\mathrm{i})} y_j^{(\mathrm{i})} = C^{(\mathrm{o})} y_j^{(\mathrm{o})}$. The eigenvalue is obtained by requiring that the solutions agree at a suitable matching point $x = x_\mathrm{f}$, say. Thus

$$\begin{aligned} C^{(\mathrm{i})} y_1^{(\mathrm{i})}(x_\mathrm{f}) &= C^{(\mathrm{o})} y_1^{(\mathrm{o})}(x_\mathrm{f}) \; , \\ C^{(\mathrm{i})} y_2^{(\mathrm{i})}(x_\mathrm{f}) &= C^{(\mathrm{o})} y_2^{(\mathrm{o})}(x_\mathrm{f}) \; . \end{aligned} \tag{3.4}$$

These equations clearly have a non-trivial solution $(C^{(\mathrm{i})}, C^{(\mathrm{o})})$ only when their determinant vanishes, *i.e.*, when

$$\Delta \equiv y_1^{(\mathrm{i})}(x_\mathrm{f}) y_2^{(\mathrm{o})}(x_\mathrm{f}) - y_2^{(\mathrm{i})}(x_\mathrm{f}) y_1^{(\mathrm{o})}(x_\mathrm{f}) = 0 \; . \tag{3.5}$$

Equation (3.5) is therefore the eigenvalue equation.

For $I = 4$ there are two linearly independent solutions satisfying the inner boundary conditions, and two linearly independent solutions satisfying the outer boundary conditions. The former set $\{y_i^{(\mathrm{i},1)}, y_i^{(\mathrm{i},2)}\}$ is chosen by setting

$$\begin{aligned} y_1^{(\mathrm{i},1)}(x_1) &= 1 \,, \qquad & y_3^{(\mathrm{i},1)}(x_1) &= 0 \,, \\ y_1^{(\mathrm{i},2)}(x_1) &= 1 \,, \qquad & y_3^{(\mathrm{i},2)}(x_1) &= 1 \,, \end{aligned} \tag{3.6}$$

and the latter set $\{y_i^{(\mathrm{o},1)}, y_i^{(\mathrm{o},2)}\}$ is chosen by setting

$$\begin{aligned} y_1^{(\mathrm{o},1)}(x_\mathrm{s}) &= 1 \,, \qquad & y_3^{(\mathrm{o},1)}(x_\mathrm{s}) &= 0 \,, \\ y_1^{(\mathrm{o},2)}(x_\mathrm{s}) &= 1 \,, \qquad & y_3^{(\mathrm{o},2)}(x_\mathrm{s}) &= 1 \,. \end{aligned} \tag{3.7}$$

The inner and outer boundary conditions are such that, given y_1 and y_3, y_2 and y_4 may be calculated from them; thus equations (3.6) and (3.7) completely specify the solutions, which may then be obtained by integrating from the inner or outer boundary. The final solution can then be represented as

$$y_j = C^{(\mathrm{i},1)} y_j^{(\mathrm{i},1)} + C^{(\mathrm{i},2)} y_j^{(\mathrm{i},2)} = C^{(\mathrm{o},1)} y_j^{(\mathrm{o},1)} + C^{(\mathrm{o},2)} y_j^{(\mathrm{o},2)} \,.$$

At the fitting point x_f continuity of the solution requires that

$$C^{(\mathrm{i},1)} y_j^{(\mathrm{i},1)}(x_\mathrm{f}) + C^{(\mathrm{i},2)} y_j^{(\mathrm{i},2)}(x_\mathrm{f}) = C^{(\mathrm{o},1)} y_j^{(\mathrm{o},1)}(x_\mathrm{f}) + C^{(\mathrm{o},2)} y_j^{(\mathrm{o},2)}(x_\mathrm{f}) \qquad j = 1, 2, 3, 4 \,. \tag{3.8}$$

This set of equations only has a non-trivial solution if

$$\Delta = \det \left\{ \begin{matrix} y_{1,\mathrm{f}}^{(\mathrm{i},1)} & y_{1,\mathrm{f}}^{(\mathrm{i},2)} & y_{1,\mathrm{f}}^{(\mathrm{o},1)} & y_{1,\mathrm{f}}^{(\mathrm{o},2)} \\ y_{2,\mathrm{f}}^{(\mathrm{i},1)} & y_{2,\mathrm{f}}^{(\mathrm{i},2)} & y_{2,\mathrm{f}}^{(\mathrm{o},1)} & y_{2,\mathrm{f}}^{(\mathrm{o},2)} \\ y_{3,\mathrm{f}}^{(\mathrm{i},1)} & y_{3,\mathrm{f}}^{(\mathrm{i},2)} & y_{3,\mathrm{f}}^{(\mathrm{o},1)} & y_{3,\mathrm{f}}^{(\mathrm{o},2)} \\ y_{4,\mathrm{f}}^{(\mathrm{i},1)} & y_{4,\mathrm{f}}^{(\mathrm{i},2)} & y_{4,\mathrm{f}}^{(\mathrm{o},1)} & y_{4,\mathrm{f}}^{(\mathrm{o},2)} \end{matrix} \right\} = 0 \,, \tag{3.9}$$

where, *e.g.*, $y_{j,\mathrm{f}}^{(\mathrm{i},1)} \equiv y_j^{(\mathrm{i},1)}(x_\mathrm{f})$. Thus equation (3.9) is the eigenvalue equation in this case.

Clearly Δ as defined in either equation (3.5) or equation (3.9) is a smooth function of σ^2, and the eigenfrequencies are found as the zeros of this function. This is done in the programme using a standard secant technique: given two values σ_{i-1}^2 and σ_i^2 of σ^2 and the associated values of Δ, the new value of σ^2 is found as

$$\sigma_{i+1}^2 = \sigma_i^2 - \Delta_i \frac{\sigma_i^2 - \sigma_{i-1}^2}{\Delta_i - \Delta_{i-1}} \,, \tag{3.10}$$

where $\Delta_j \equiv \Delta(\sigma_j^2)$. However, the programme also has the option for scanning through a given interval in σ^2 to look for change of sign of Δ, possibly iterating for the eigenfrequency at each change of sign. Thus it is possible to search a given region of the spectrum completely automatically.

The programme allows the use of two different techniques for solving the differential equations. One is the standard second-order centred difference technique, where the differential equations are replaced by the difference equations

$$\frac{y_i^{n+1} - y_i^n}{x^{n+1} - x^n} = \frac{1}{2} \sum_{j=1}^{I} \left[a_{ij}^n y_j^n + a_{ij}^{n+1} y_j^{n+1} \right] \,, \quad i = 1, \ldots, I \,. \tag{3.11}$$

Here we have introduced a mesh $x_1 = x^1 < x^2 < \cdots < x^N = x_s$ in x, where N is the total number of mesh points; $y_i^n \equiv y_i(x^n)$, and $a_{ij}^n \equiv a_{ij}(x^n)$. These equations allow the solution at $x = x^{n+1}$ to be determined from the solution at $x = x^n$.

The second technique was proposed by Gabriel & Noels (1976); here on each mesh interval (x^n, x^{n+1}) we consider the equations

$$\frac{\mathrm{d}y_i^{(n)}}{\mathrm{d}x} = \sum_{j=1}^{I} \bar{a}_{ij}^n y_j^{(n)}(x), \qquad \text{for } i = 1\,, \ldots, I\,, \tag{3.12}$$

with constant coefficients, where $\bar{a}_{ij}^n \equiv 1/2(a_{ij}^n + a_{ij}^{n+1})$. These equations may be solved analytically on the mesh intervals, and the complete solution is obtained by continuous matching at the mesh points. This technique clearly permits the computation of solutions varying arbitrarily rapidly, *i.e.*, the calculation of modes of arbitrarily high order. On the other hand solving equations (3.12) involves finding the eigenvalues and eigenvectors of the coefficient matrix, and therefore becomes very complex and time consuming for higher-order systems. Thus in practice it has only been implemented for systems of order 2, *i.e.*, radial oscillations or non-radial oscillations in the Cowling approximation.

3.2 The relaxation technique

Two variants of this technique have been implemented. In the first one of the boundary conditions (to be definite we assume here that it is the first surface condition) is set aside to be used for determining the eigenfrequency. The difference equations (3.11) for $n = 1, 2, \ldots, N-1$, the boundary conditions (3.2), the normalization condition $y_1(x_s) = 1$, and, for $I = 4$, the remaining surface boundary condition, are then solved to give the solution $\{y_i^n\}$ at each mesh point. Notice that the equations and boundary conditions constitute a set of linear equations for the solution, and this set may be solved efficiently by forward elimination and backsubstitution (*e.g.* Baker, Moore & Spiegel 1974). The eigenvalue is then found by requiring that the remaining boundary condition be satisfied; thus we evaluate

$$\Delta = \frac{\sum_{j=1}^{I} c_{1j} y_j(x_s)}{\left[\sum_{j=1}^{I} \left(y_j^{(\mathrm{norm})}\right)^2\right]^{1/2}}\,, \tag{3.13}$$

where $y_j^{(\mathrm{norm})} = y_j(x^{(\mathrm{norm})})$ is the solution at a suitably chosen normalization point; in the present case, where a surface boundary condition is used, $x^{(\mathrm{norm})}$ is typically the innermost mesh point (the added flexibility of being able to vary $x^{(\mathrm{norm})}$ has occasionally been found to be useful). Δ is then a smooth function of σ^2, and the eigenvalues may be found as the zeros of Δ, essentially as described in connection with the shooting technique.

As both boundaries, at least in a complete model, are either singular or very nearly singular, the removal of one of the boundary conditions tends to produce solutions that are somewhat ill-behaved, in particular for modes of high degree. This in turn is reflected in the behaviour of Δ as a function of σ^2. This problem is avoided in the second variant of the relaxation technique. Here the difference equations are solved separately for $x \leq x_\mathrm{f}$ and $x \geq x_\mathrm{f}$, by introducing a double point $x_\mathrm{f}^- = x^{n_f} = x^{n_f+1} = x_\mathrm{f}^+$ in the mesh. The solution is furthermore required to satisfy the boundary conditions (3.2) and (3.3), a suitable normalization condition (*e.g.* $y_1(x_s) = 1$), and continuity of all but one of the variables at $x = x_\mathrm{f}$, *i.e.*,

$$\begin{aligned} y_1(x_\mathrm{f}^-) &= y_1(x_\mathrm{f}^+)\,, \\ y_3(x_\mathrm{f}^-) &= y_3(x_\mathrm{f}^+)\,, \\ y_4(x_\mathrm{f}^-) &= y_4(x_\mathrm{f}^+)\,, \end{aligned} \tag{3.14}$$

(when $I = 2$ clearly only the first continuity condition is used) We then set

$$\Delta = y_2(x_f^-) - y_2(x_f^+) , \qquad (3.15)$$

and the eigenvalues are found as the zeros of Δ, regarded as a function of σ^2. It should be noticed, however, that with this definition, Δ may have singularities with discontinuous sign changes that are not associated with an eigenvalue. Assuming that the normalization is at the outer boundary, as used above, this occurs when $y_1(x_f^-)$ is very close to a zero. Then the continuous fitting to $y_1(x_f^+)$, which is in general not close to a zero, forces the interior solution, and hence Δ, to be very large. Thus care is needed when searching for the eigenvalues by stepping through a range in σ^2, when this method is used. However, close to an eigenvalue Δ is generally well-behaved, and the secant iteration (*cf.* equation 3.10) may be used without problems.

There is a third variant of the relaxation technique, which has not, however, been implemented in the current version of the programme (*cf.* Unno *et al.* 1989, pp. 167 – 178). Here the difference equations, the boundary conditions and a normalization condition are solved simultaneously for the solution $\{y_i^n\}$ and the eigenvalue σ^2. As the eigenvalue is included as an unknown when solving the equations, the system is non-linear and must be solved by Newton-Raphson iteration. This method in principle gives quadratic convergence towards the solution (provided analytical derivatives of the equations and boundary conditions with respect to the y_i^n and σ^2 are used), as opposed to the somewhat slower convergence obtained with the secant iteration. However, it requires an initial guess for both the eigenvalue and the eigenfunction; experience with this method in a different programme suggests that this guess has to be fairly close to the correct solution, at least for modes that are predominantly trapped in a restricted region of the model. Furthermore there is no natural way to scan the spectrum. Thus this method in practice almost certainly has to be combined with a version of one of the methods discussed above, to provide an initial estimate of the eigenvalue and eigenfunction. It could then be used with some advantage for the final iterations towards the solution, where its faster convergence could be exploited. This would be a fairly straightforward extension of the existing programme, and may be implemented in future.

3.3 The transition from $\hat{y}_i$ to y_i

In practice, the variables $\hat{y}_i$ are defined as

$$\hat{y}_i(x) = \left(\frac{x}{x_{\rm ev}}\right)^{-l+1} y_i(x) , \qquad (3.16)$$

rather than by equation (2.6). Also the transition point $x_{\rm ev}$ is taken at an existing point in the mesh. To simplify the logic in the programme $x_{\rm ev}$ is forced to be closer to the centre than the fitting point $x_{\rm f}$; if $x_{\rm f}$ has been specified such that this puts constraints on $x_{\rm ev}$, a warning is printed (note that, as discussed in Section 3.5 below, the fitting point should not be placed in the evanescent region).

From the definition (3.16) it follows that

$$\hat{y}_i(x_{\rm ev}) = y_i(x_{\rm ev}) . \qquad (3.17)$$

When using the shooting technique the integration is carried out separately for x between x_1 and $x_{\rm ev}$, and x between $x_{\rm ev}$ and $x_{\rm f}$, and equations (3.17) are used to provide initial conditions for the integration on the latter region. When the relaxation technique is used, the integration is carried out over the entire region between x_1 and $x_{\rm f}$; to ensure that the conditions (3.17) are satisfied a double point $(x_{\rm ev}^-, x_{\rm ev}^+)$ is introduced where $x_{\rm ev}^- = x_{\rm ev}$, $x_{\rm ev}^+ = x_{\rm ev} + \epsilon$, and ϵ should be chosen close to machine accuracy; in this way the difference equations (3.11) approximately ensures that equations (3.17) are satisfied. It would be possible, at the price of making the programme somewhat more complicated, to enforce equations (3.17) strictly, but this is unlikely to have any significant effect on the results.

3.4 Richardson extrapolation

The difference scheme (3.11), which is used by one version of the shooting technique, and the relaxation technique, is of second order. Consequently the truncation errors in the eigenfrequency and eigenfunction scale as N^{-2}. If $\omega(\frac{1}{2}N)$ and $\omega(N)$ are the eigenfrequencies obtained from solutions with $\frac{1}{2}N$ and N meshpoints, the leading order error term therefore cancels in

$$\sigma_{\rm Ri} = \frac{1}{3}[4\sigma(N) - \sigma(\frac{1}{2}N)] \; . \tag{3.18}$$

This procedure, known as *Richardson extrapolation*, was used by Shibahashi & Osaki (1981). It provides an estimate of the eigenfrequency that is substantially more accurate than $\sigma(N)$, although of course at some added computational expense.

It is obviously essential that the iteration converges to the same mode in the calculations with N and $N/2$ meshpoints. To check this, the order of the solution obtained in the two cases is calculated, according to the procedure described in equation (4.1) below. If the two solutions have different orders, the combined solution should in principle be rejected. In practice, the situation is sometimes more complex, due to problems with the definition of the order. It happens, particularly for $l = 1$ in evolved models, that the computed order jumps during evolution, due to a slight shift in the eigenfunction which eliminates a zero. Similarly, corresponding differences between the eigenfunctions computed with $N/2$ and N points can lead to different orders being inferred in the two cases, even though this is in fact the same mode, and Richardson extrapolation therefore justified. This is handled by the programme in the following way: if different orders are detected, the difference between the eigenfunctions is estimated, in terms of the norm defined by the energy integral (equation 4.3), relative to the norm of one of the eigenfunctions. If this norm is comparable with the difference between the frequencies, in a suitable sense, the mode is accepted, otherwise it is rejected. In the former case a warning message, in the latter an error message, is written to `adipls-status.log`. The details of the tests (which are handled by the subroutine `testri`) may still have to be fine-tuned.

3.5 Operational experience

As implemented here the shooting technique is considerably faster than the relaxation technique, and so should be used whenever possible (notice that both techniques may use the difference equations (3.11) and so they are numerically equivalent, in regions of the spectrum where they both work). For *second-order systems* the shooting technique can probably always be used; the integrations of the inner and outer solutions should cause no problems, and the matching determinant in equation (3.5) is well-behaved. For *fourth-order systems*, however, this need not be the case. For modes where the perturbation in the gravitational potential has little effect on the solution, the two solutions $y_j^{(\rm i,1)}$ and $y_j^{(\rm i,2)}$, and similarly the two solutions $y_j^{(\rm o,1)}$ and $y_j^{(\rm o,2)}$, are almost linearly dependent, and so the matching determinant nearly vanishes for any value of σ^2. This is therefore the situation where the relaxation technique may be used with advantage. The point where the shooting method fails depends on the precision. With 4-byte real variables, such as is used in the single-precision version on the Aarhus Alliant, experience has shown that for a solar model the shooting technique may give problems for the full fourth order system for 5 min modes of degree higher than 4. On the NCAR Cray, with 8 byte real variables, it was possible to go as high as l = 40 for 5 min solar p modes; for strongly trapped g modes there were problems at degree higher than about 15. However, in general a little experimentation may be required to determine the conditions under which one method should be preferred over another.

It should be noticed that the conditions under which the shooting technique for the full system gives problems are precisely the conditions where the Cowling approximation (of neglecting the perturbation in the gravitational potential) might be expected to be applicable. Thus here the Cowling approximation may be adequate, and the resulting second-order system can be solved using the shooting technique; the resulting eigenfrequency may, if desired, be corrected for the effects of the perturbations in the gravitational potential by using Cowling's perturbation expression, and the perturbation in the potential may be computed from the eigenfunction by using the integral solution of Poisson's equation (this option is built into the programme). If a solution of the full system is desired, a reasonable strategy is to first compute the solution in the Cowling approximation with the shooting technique (possibly stepping in σ^2 to search for the eigenvalues), and to then solve the full set using the relaxation technique, starting from the eigenvalues found in the Cowling approximation.

The ability to specify the fitting point $x_{\rm f}$ gives considerably flexibility in the calculation, and correct choice of $x_{\rm f}$ is often essential to obtain a solution. As a rule of thumb when using the shooting technique $x_{\rm f}$ should be chosen near the general maximum in the eigenfunction, so that the integration both from the inner and from the outer boundary is in a direction where the solution is predominantly increasing. However, the fitting point may not placed too close to the surface, as otherwise the almost singular solution at the surface may cause problems.

Problems are almost certain to arise if the fitting point is placed deep within the region where the solution is evanescent. In particular, in older versions of the code the point $x_{\rm ev}$ where the transition from scaled to unscaled variables takes place was restricted to lie deeper than the fitting point. For very high-degree modes, where the turning point is within a fraction of a per cent of the radius from the surface, this may easily happen, unless $x_{\rm f}$ is very close to unity. In the present version of the code, this can be controlled in two slightly different ways:

i) if the parameter `irsevn` is set to 2, the fitting point is automatically reset to $x_{\rm ev}$ if it is found that $x_{\rm f} < x_{\rm ev}$. The fitting point is reset to the input value before the next mode is calculated.
ii) if the parameter `xfit` is set to -1, the fitting point is always set to the boundary of the evanescent region.

Roughly the same rules apply when using the relaxation technique with fitting, although here the rate (or even success) of convergence appears to somewhat more sensitive to the proper choice of $x_{\rm f}$. It has been found that the programme occasionally fails to converge to a single mode, out of a large sample or converges to a neighbouring mode. The missing mode can then usually be found by choosing a slightly different value for $x_{\rm f}$. The programme has the option of automatically resetting $x_{\rm f}$ if the iteration fails or the wrong mode is obtained, by setting the parameter `nftmax` to a value greater than one. If this succeeds, a warning message is written to `adipls-status.log`. After completing the given mode, $x_{\rm f}$ is reset to the original value.

(As indicated previously the use of the relaxation technique, removing one of the boundary conditions, can cause problems; in particular it seems to give trouble for precisely the modes where the relaxation technique is indicated, *i.e.*, modes of high degree. This option was implemented before the relaxation technique with interior matching, and has been left in the programme to allow possible future experiments with its properties; however, for practical calculations the relaxation technique with interior matching should be preferred.)

4 Results of the calculation

The principal results of the calculation are the eigenfrequency σ and the eigenfunction $y_i(x)$. However, the programme computes and prints a number of other quantities, which are described in this section.

4.1 Frequency correction in the Cowling approximation

When the equations are solved in the Cowling approximation for a complete model, the correction $\delta\sigma^2$ to σ^2 caused by the perturbation in the gravitational potential is calculated from Cowling's (1941) perturbation expression, as well as the corrected squared frequency $\sigma_c^2 = \sigma^2 + \delta\sigma^2$; here σ^2 denotes the value obtained as an eigenvalue. The calculation is carried out by the subroutine `gravpo`. In addition the perturbation of the gravitational potential, in the form of y_3 is calculated from the integral solution to Poisson's equation; currently y_4 is not set.

4.2 Mode order

The programme finds the order of the mode according to the Scuflaire (1974) definition (see also Unno *et al.* 1989, p. 149 – 158), through calling subroutine `order`. Specifically the order is defined by

$$n = - \sum_{x_{z1} > 0} \text{sign}\left(y_2 \frac{\mathrm{d}y_1}{\mathrm{d}x} \right) + n_0 \; . \qquad (4.1)$$

Here the sum is over the zeros $\{x_{z1}\}$ in y_1 (excluding the centre), and sign is the sign function, $\text{sign}(z) = 1$ if $z > 0$ and $\text{sign}(z) = -1$ if $z < 0$. The value of n_0 depends on the behaviour of the solution close to the innermost boundary: if y_1 and y_2 have the same sign at the innermost mesh point, excluding the centre, $n_0 = 0$, otherwise $n_0 = 1$. In particular, for a complete model that includes the centre, it follows from the expansion of the solution at the centre that $n_0 = 1$ for radial oscillations and $n_0 = 0$ for non-radial oscillations. Thus the fundamental radial oscillation has order $n = 1$. Although this is contrary to the commonly used convention of assigning order 0 to the fundamental radial oscillation, the convention used here is in fact the more reasonable, in the sense that it ensures that n is invariant under a continuous variation of l from 0 to 1. With this definition $n > 0$ for p modes, $n = 0$ for f modes, and $n < 0$ for g modes.

It may be shown that this labelling is mathematically satisfactory in the Cowling approximation, in the sense of being invariant under continuous variations of the equilibrium model, or under continuous changes in l (Gabriel & Scuflaire 1979; Christensen-Dalsgaard 1980). However, this is not always the case for modes corresponding to the full set of equations. In particular, for sufficiently centrally condensed models an unreasonable order is calculated for the lowest-order p modes with $l = 1$; such models include models of the present Sun and more highly evolved models, as well as polytropic models of sufficiently high polytropic index (see Christensen-Dalsgaard & Mullan 1994). The correct mode labelling can in principle be inferred for such models by following the modes through a sequence of models starting, *e.g.*, from a ZAMS model where such problems apparently do not occur.

The programme contains the option for correcting the order set as in equation (4.1), controlled by the input parameter `irsord`: if $1 \leq$ `irsord` ≤ 10, the order n as calculated using equation (4.1) is replaced by $\tilde{n} = n + 1$ when $l = 1$ and $0 \leq n \leq$ `irsord`. Thus `irsord = 1` makes the correct resetting in the case of a traditional model of the present Sun. It is important to realize, however, that the correction depends strongly on the nature of the model. In particular, for more evolved $1\,\mathrm{M}_\odot$ models the problems extend to higher order.

An alternative method for setting the order was proposed by Lee (1985), on the basis of $\delta\Phi$ and p'. It may be obtained in the programme by setting `irsord = 11` or `-11`. However, the success with this approach has not been entirely convincing so far.

4.3 Mode energy, scaled eigenfunctions

The programme calls subroutine `ekin` to calculate a dimensionless measure of the kinetic energy of pulsation. Two different normalizations may be chosen, depending on the value of the parameter `iekinr`. For `iekinr` = 0 (the default) the quantity calculated is

$$E = \frac{\int_{r_1}^{R_s} [\xi_r^2 + l(l+1)\xi_h^2] \rho r^2 \mathrm{d}r}{M \xi_r(R_s)^2} = \frac{\displaystyle\int_{x_1}^{x_s} \left[y_1^2 + y_2^2/l(l+1)\right] qU \mathrm{d}x/x}{4\pi y_1(x_s)^2} , \qquad (4.2a)$$

whereas for `iekinr` = 1,

$$E = \frac{\int_{r_1}^{R_s} [\xi_r^2 + l(l+1)\xi_h^2] \rho r^2 \mathrm{d}r}{M[\xi_r(R_{\rm phot})^2 + l(l+1)\xi_h(R_{\rm phot})^2]} = \frac{\int_{x_1}^{x_s} \left[y_1^2 + y_2^2/l(l+1)\right] qU \mathrm{d}x/x}{4\pi[y_1(x_{\rm phot})^2 + y_2(x_{\rm phot})^2/l(l+1)]} . \qquad (4.2b)$$

(For radial modes the terms in y_2 are not included.) Here $r_1 \equiv R x_1$, $R_s \equiv R x_s$ and $R_{\rm phot} \equiv R$ are the distance of the innermost mesh point from the centre and the surface and photospheric radii of the model, respectively; $q = m/M$ and $U = 4\pi \rho r^3/m$.

To indicate the region where the mode predominantly resides (in an energetical sense)

$$\begin{aligned} z_1(x) &= \left(\frac{4\pi r^3 \rho}{M}\right)^{1/2} y_1(x) = \left(\frac{4\pi r^3 \rho}{M}\right)^{1/2} \frac{\xi_r(r)}{R} , \\ z_2(x) &= \frac{1}{\sqrt{l(l+1)}} \left(\frac{4\pi r^3 \rho}{M}\right)^{1/2} y_2(x) = \sqrt{l(l+1)} \left(\frac{4\pi r^3 \rho}{M}\right)^{1/2} \frac{\xi_h(r)}{R} , \end{aligned} \qquad (4.3)$$

are calculated (for radial modes only z_1 is found). These are defined in such a way that

$$E = \frac{1}{4\pi y_1(x_s)^2} \int_{x_1}^{x_s} [z_1^2 + z_2^2] \frac{\mathrm{d}x}{x} \qquad (4.4)$$

[assuming the definition (4.2a) of E]. For output purposes the normalized functions

$$\hat{z}_1 = \frac{z_1}{z_{1,\rm max}} \quad \text{and} \quad \hat{z}_2 = \frac{z_2}{z_{1,\rm max}} , \qquad (4.5)$$

where $z_{1,\rm max}$ is the maximum value of z_1, are generally used.

4.4 Variational frequency

The programme has the option of calculating the frequency and period based on the variational expression for ω^2, in the subroutine `varfrq` [see Christensen-Dalsgaard (1982) for details]. As discussed there, different formulations have to be used for p and for g modes, selected by the input parameter `ivarf`. It might also be noticed that, as implemented, the calculation is based on only y_1 and y_2, with the contribution from the perturbation in the gravitational potential being calculated using the integral solution to Poisson's equation. Thus even for modes calculated in the Cowling approximation the perturbation of the gravitational potential is taken into account, essentially by using the perturbation technique, in the calculation of the variational period.

Notice that the programme normally uses a different formulation for radial and for nonradial modes. There is an option (selected by setting the parameter `ivarf` to 3), where essentially the nonradial formulation of the variational integral (in its p-mode form) may be used for radial modes; this may be useful to get strict consistency between the calculation for radial and non-radial modes. However, this formulation apparently leads to fairly severe cancellation and consequent loss of accuracy. Thus it probably cannot be recommended, although the modes for which it might be of use must be investigated.

4.5 Kernels

For a rotation law depending on r only the rotational splitting may be written

$$\Delta\omega_{nlm} = m\beta_{nl} \int_0^{x_s} K_{nl}(x)\Omega(x)\mathrm{d}x \; , \tag{4.6}$$

where $\Omega(x)$ is the angular velocity. The programme has an option of calculating and printing β_{nl} and K_{nl}; this is done by subroutine `rotker` when the parameter `irotkr` is set to 1.

The programme contains also the option for computing the kernel for computing the frequency change caused by a change in Γ_1 at fixed p and ρ. Specifically, the kernel $K_{nl}^{(\Gamma_1)}$ is defined such that

$$\frac{\delta\omega}{\omega} = \int_0^{x_s} K_{nl}^{(\Gamma_1)} \delta\Gamma_1 \mathrm{d}x \; . \tag{4.7}$$

It is calculated by subroutine `gm1ker` when the parameter `igm1kr` is set to 1.

5 Equilibrium model variables

The following variables are needed at each mesh point in the model:

$$\begin{aligned}
& x \equiv r/R \; , \\
& A_1 \equiv q/x^3, \qquad \text{where } q = m/M \; , \\
& A_2 = V_g \equiv -\frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} = \frac{Gm\rho}{\Gamma_1 p r} \; , \\
& A_3 \equiv \Gamma_1 \; , \\
& A_4 = A \equiv \frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} - \frac{\mathrm{d}\ln\rho}{\mathrm{d}\ln r} \; , \\
& A_5 = U \equiv \frac{4\pi\rho r^3}{m} \; .
\end{aligned} \tag{5.1}$$

These are clearly all dimensionless. In addition we use the following "global" quantities for the model:

$$\begin{aligned}
& D_1 \equiv M \; , \\
& D_2 \equiv R \; , \\
& D_3 \equiv p_\mathrm{c} \; , \\
& D_4 \equiv \rho_\mathrm{c} \; , \\
& D_5 \equiv -\left(\frac{1}{\Gamma_1 p}\frac{\mathrm{d}^2 p}{\mathrm{d}x^2}\right)_\mathrm{c} \; , \\
& D_6 \equiv -\left(\frac{1}{\rho}\frac{\mathrm{d}^2 \rho}{\mathrm{d}x^2}\right)_\mathrm{c} \; , \\
& D_7 \equiv \mu \; , \\
& D_8 : \text{see below} \; .
\end{aligned} \tag{5.2}$$

Here R and M are photospheric radius and mass of the model (the photosphere being defined as the point where the temperature equals the effective temperature). In a complete model p_c and ρ_c are central pressure and density, and D_5 and D_6 are evaluated at the centre. In an envelope model (that does not include the centre) D_3 and D_4 should be set to the values of pressure and density at the innermost mesh point, and D_5 and D_6 may be set to zero. The dimensional variables (*i.e.*, $D_1 - D_4$) must be given in *cgs* units.

For a model with vanishing surface pressure μ is the effective polytropic index at the surface, so that in particular μ is the polytropic index of a complete polytrope (the polytropic index elsewhere is not needed in the calculation, and so mixed polytropes can be considered); models with non-vanishing surface pressure are flagged by having $\mu < 0$. The notation is otherwise standard. The model may include an atmosphere (for solar models a simplified atmosphere extending out to roughly the temperature minimum is typically used). Thus at the surface possibly $x > 1$.

These variables are convenient when the equations are formulated as by *e.g.* Dziembowski; but it should be possible to derive any set of variables required for *adiabatic* oscillation calculations from them. D_5 and D_6 are needed for the expansion of the solution around the centre. In models with vanishing surface pressure D_7 is used in the expansion at the surface singular point.

For a plane-parallel equilibrium model, or when the equilibrium model is affected by turbulent pressure, the definition of the equilibrium variables must be somewhat modified. These modifications are discussed in Sections A.2 and A.3, respectively.

The quantity D_8 is used to flag for a different number of variables in the file, or otherwise a different structure. Currently the only non-standard option is to have $D_8 = 10$. In this case, the file contains 6 variables $A_1 - A_6$ at each mesh point, A_6 being related to the gradient in turbulent pressure (see Section A.3.2). The value of D_8 is checked when the file is opened in subroutine `readml`; hence the structure must be the same for all models in a given file.

The following relations between the variables defined here and more “physical” variables are often useful:

$$p = \frac{GM^2}{4\pi R^4}\frac{x^2 A_1^2 A_5}{A_2 A_3} , \qquad \frac{\mathrm{d}p}{\mathrm{d}r} = -\frac{GM^2}{4\pi R^5} x A_1^2 A_5 , \qquad \rho = \frac{M}{4\pi R^3} A_1 A_5 . \tag{5.3}$$

We may also express the characteristic frequencies for adiabatic oscillations in terms of these variables. Thus if N is the buoyancy frequency, S_l is the Lamb frequency and ω_a is the acoustical cut-off frequency for an isothermal atmosphere, we have

$$N^2 \equiv \frac{GM}{R^3}\tilde{N}^2 = \frac{GM}{R^3} A_1 A_4 , \tag{5.4}$$

$$S_l^2 \equiv \frac{l(l+1)c^2}{r^2} \equiv \frac{GM}{R^3}\tilde{S}_l^2 = \frac{GM}{R^3}\frac{l(l+1)A_1}{A_2} , \tag{5.5}$$

and

$$\omega_a^2 \equiv \frac{c^2}{4H_p^2} = \frac{1}{4}\frac{GM}{R^3} A_1 A_2 A_3^2 , \tag{5.6}$$

where c is the adiabatic sound speed, and $H_p = p/(g\rho)$ is the pressure scale height. Finally it may be noted that the squared sound speed is given by

$$c^2 = \frac{GM}{R} x^2 \frac{A_1}{A_2} . \tag{5.7}$$

6 Notation and data storage in the programme

6.1 Model storage

Only a limited description of the notation in the programme is given in this section, aimed at facilitating the later discussion of input to and output from the programme. Further comments on the organization of the programme are given in Appendix B. However, a detailed description of the code is outside the scope of these notes.

In the programme a mesh $\{x^n\}$ is used, with x^1 being the point closest to the centre and x^{NN} at the surface; this is stored in a one-dimensional array $\texttt{x(n)} = x^n, n = 1, 2, \ldots, NN$. The A_i are stored in the array `aa(1:iaa,1:nn)`. Here the first dimension `iaa` is currently set to 10. The variables are stored as

$$\texttt{aa(i,n)} = A_i(x^n), \qquad i = 1, \ldots, \texttt{ia}, \quad n = 1, \ldots, \texttt{nn} \, ,$$

where `ia` is normally 5, but may be 6 in models including one possible treatment of turbulent pressure (*cf.* section A.3.2); `aa(10,n)` is used for additional storage related to a different way of handling turbulent pressure. Finally the D_i are stored in the array `data(1:8)` as

$$\texttt{data(i)} = D_i, \qquad i = 1, \ldots, 7 \, ;$$

`data(8)` is used only as a flag on model input (*cf.* Section 5).

6.2 Storage of solution

The *squared* eigenfrequency σ^2 is denoted by `sig` (and `sig1`, `sig2`, `sigp`, ...) in the programme. Furthermore the degree l of the modes is stored in `el`; as discussed in Section 2.1 l is treated as a real variable.

The solution at `x(n)` is set into `y(i,n)`, as

$$\texttt{y(i,n)} = y_i(x^n), \quad \texttt{i} = 1, \ldots, \texttt{ii}, \quad \texttt{n} = \texttt{nw1}, \ldots, \texttt{nn} \, .$$

Here `ii` is the order of the system, *i.e.*, 2 for radial oscillations or non-radial oscillations in the Cowling approximation, and 4 for non-radial oscillations with the full system. Further `nw1` = 1 if the model is not truncated in the interior, otherwise `nw1 = ntrnct`, the truncation mesh point; `nnw = nn - nw1 + 1` is used for the total number of points in the solution.

The inner boundary conditions are applied at the point `nibc`. When `nw1` = 1, and `x(1)` is at the centre of the model, `nibc` = 2. Otherwise `nibc = nw1`.

A more detailed description of the data storage is given in the source for the programme.

6.3 Terminal input, and terminal and printed output

The unit numbers for these types of input and output are defined in

```
        common/cstdio/ istdin, istdou, istdpr
```

Here `istdin` is the unit number for input (typically 5), `istdou` is the unit number for output to the terminal, and `istdpr` is the unit number for printed output. For batch processing, one would obviously have `istdou = istdpr`. These quantities are initialized in

```
      block data blstio
```

which must be set, depending on the installation. `istdpr` is contained in the list of control parameters, and may be changed during the operation of the programme. This allows, say, sending a short summary of the operation of the programme, to verify that it is successful, to the terminal, and the detailed printed output to a file.

7 Input to the programme

7.1 The equilibrium model

The equilibrium model must be supplied to the programme in a single, unformatted record, to be read by the statement

```
       read(imlds) nmod, nnmodl, (data(i), i=1,8), (x(n), (aa(i,n), i=1,ia),
      * n=1,nnmodl)
```

Here `nmod` is an identification number of the model, which is not used in the calculation. Also, `ia` should be 5, unless anything else is flagged by setting `data(8)`. Note that the model storage must start at the centre.

It is important that the model supplied to the programme must have a reasonable distribution of mesh points. There is currently no possibility for resetting the mesh in the programme, apart from possibly reducing the number of points by taking every `in`-th point (*cf.* Section 7.2). This is of course particularly important for high-order modes, where it is essential to use different meshes for p and for g modes. A separate programme exists for resetting the mesh from that supplied by, *e.g.*, the evolution programme.

7.2 Control parameters

7.2.1 Input of control parameters

In the original version of the programme control parameters were input by means of `namelist`. However, `namelist` is not a standard Fortran77 feature, unfortunately; in particular, certain implementations under UNIX have not permitted `namelist`. Also `namelist` may not be optimal for interactive use of the programme.

Thus the programme uses list-directed input. This takes place in separate lines, containing up to 9 variables. On *input*, each line is prompted by printing the names of the variable, and the current values, as in

```
      itrsig, sig1, istsig, inomde, itrds ?
      0 10.00 1 1 10
```

The new values are then input. Note that if a value is not given for a variable, it is not changed. Thus by inputting, *e.g.*,

```
      ,5.,,2
```

in response, `sig1` is set to 5.0, `inomde` to 2, and the remaining variables are unchanged. If a line consisting solely of commas is input, all variables are unchanged.

Although it is possible in this way to enter the data from the terminal, this is somewhat impractical, given the large number of parameters. The usual way of providing input to the code is through the use of an input file. This may conveniently be given the following structure:

```
    .
    .
mod:
ifind, xmod, imlds, in, nprmod,
,,,,,,, @
xtrnct, ntrnsf, imdmod,
0.,,,,,,,,,,,,,,,,,,,,,, @
osc:
el, nsel, els1, dels, dfsig1, dfsig2, nsig1, nsig2
,4,0,1,0,0,,,,,,, @
itrsig, sig1, istsig, inomde, itrds,
1, 5 0 , 10,10,,,,,,,, @
dfsig, nsig, iscan, sig2,
,2,50,30,,,,,,,,,,,,,,,,,,,,,,, @
eltrw1, eltrw2, sgtrw1, sgtrw2
,,0,-1,,,,,,,,,,,,,,,,,,,,,,,,,,,, @
    .
    .
```

Here lines of headers giving variable names are followed by lines setting their values. The latter are indicated by the character "@" at the end of the line. Only these data lines should be passed to the programme. This is achieved by passing the input file through a filter, before the data are read in.* In practice the extraction is handled automatically by the UNIX script which is invoked to run the programme (*cf. Notes on using the solar models and adiabatic pulsations package*).

The *output* gives a line with the variable names, and a line giving their values, as

```
itrsig, sig1, istsig, inomde, itrds
0 5.0 1 2 10 @
```

Note the addition of the character "@". This is included in all output lines that correspond to a parameter input line to the programme; in this way, the output contains essentially the information required to repeat the given run.

As the programme has a very large number of control parameters (about 85!), these are separated into 5 groups. The first control parameter, `cntrd`, is a string which controls which of these groups are modified. Each group is indicated by a three-letter string, defined by

`mod`: read model controls.
`osc`: read controls for mode selection.
`cst`: read new values of fundamental constants (currently just gravitational constant)
`int`: read controls for type of equations or integration procedure.
`out`: read controls for output.
`dgn`: read controls for diagnostics.

* This extraction is trivially done under UNIX by using the `grep` programme. More generally, it would be simple to write a Fortran programme to perform the same task.

The fields may be separated by, e.g., "." (but *not* comma or blank, as they act as variable separators in the list-directed input). Thus if `cntrd` is "`mod.osc.int`", the groups defining the model, mode selection and integration procedures are read. Note that this input quantity *must* be specified explicitly (*i.e.*, it cannot be set to default values by means of ",,,").

7.2.2 Assignment of unit numbers to files

In the current version of the programme, files are accessed by unit numbers which are assigned once and for all. This is therefore very similar to the procedure used on traditional main-frames, where the assignment takes place with the appropriate JCL statements. – In future a more flexible scheme may be implemented.

The assignment takes place in the programme, to make it as installation independent as possible. It is defined by means of a list, read from normal input, on the form

```
<unit number> <file name>
```

and terminated by

```
-1 ''
```

As an example, the following block of an input file

```
2 'model'     @
10 '/usr/jcd/agsm/gsm'     @
11 'new.gsm'     @
15 'new.ssm'     @
-1 ''     @
```

assigns unit number 2 to the file "`model`", unit number 10 to the file `/usr/jcd/agsm/gsm`, and so on (note that the " ' " are required here to ensure input of the entire string; the file name format is peculiar to UNIX). Notice the addition of "`@`" to make these into input lines passed to the programme.

The programme prompts for input of the file data, giving information about the files required and the format. The input data is echoed, with the addition to each line of the character "`@`", in accordance with the usage described above in connection with list-directed input/output.

7.2.3 Description of control parameters

Below is a complete list of the control variables and their meaning. This is essentially copied from the list given in the programme. To facilitate the use of the programme a discussion of some of the options is given in Sections 7.3 – 7.5.

Note that to separate the variables in the programme from the other variables, the former have been written in upper case. In fact the current version of the programme is written entirely in lower case.

`cntrd`: String determining which control fields are read. See description in Section 7.2.2 above.
(default: `cntrd = 'mod.osc.int.out.dgn'`. This reads all the control fields, except fundamental constants.)

a) Equilibrium model controls (group `mod`):

`ifind`, `xmod`: determines which model is used.

`ifind` $<$ 0: do not read new model, except if no model has been read so far.

`ifind` $= -2$: possibly reset model with subroutine `modmod` or set new truncation, even if no new model was read.

`ifind` $= 0$: rewind data set before reading model.

`ifind` $= 1$: read next model on data set.

`ifind` $= 2$: read model no. `xmod` on dataset.

`xmod` is currently assumed be integer-valued. Non-integer `xmod` may be implemented later, with interpolation between models.

(default: `ifind = -1`
`xmod = 0`)

`imlds`: models are read from unit `imlds`

(default: `imlds = 2`)

`in`: after reading model, take every `in`-th point

(default: `in = 1`)

`nprmod`: if `nprmod` $>$ 0 print the model at `nprmod` points.

(default: `nprmod = 0`)

`xtrnct`: if `xtrnct` $>$ 0 truncate the model fractional radius $r/R =$ `xtrnct`. This is only implemented for radial oscillations, or in the Cowling approximation. Thus if $l > 0$ and the model is truncated, `icow` $= 2$ is forced, and a diagnostics is printed.

(default: `xtrnct = 0`)

`ntrnsf`: if `ntrnsf` $>$ 1 stop model at point no. `ntrnsf` from the surface.

(default: `ntrnsf = 0`)

`imdmod`: when `imdmod` $\neq 0$ call `s/r modmod`, which may be user-specified to modify the model.

(default: `imdmod = 0`)

b) Controls for l and the trial frequency (group `osc`):

`el`, `nsel`, `els1`, `dels`: controls for determining the value of the degree l.

`el`: when `nsel` ≤ 0 (and `itrsig` $\neq 2$; see below) use $l =$ `el` as input.

(default: `el = 1`)

`nsel`: when `nsel` ≥ 1 step through $l =$ `els1` $+ i *$ `dels`, $i = 0, \ldots,$ `nsel` $- 1$. When `iscan` $\leq$ 1 (see below) `sig` is given the initial value in `sig1`. The increment from one value of l to the next may be controlled by the parameters `dfsig1, nsig1` (see below).

(default: `nsel = 0`
`els1 = 10`
`dels = 1`)

`dfsig1`, `dfsig2`, `nsig1`, `nsig2`: allows resetting of the `sig1` and `sig2` during step in l for `nsel` > 1. For each new value of l, `sig1` and `sig2` are incremented as determined by (`nsig1, dfsig1`), (`nsig2, dfsig2`), in the same manner as in the definition of `nsig` and `dfsig` below.

(default: `dfsig1 = 0`
`dfsig2 = 0`
`nsig1 = 1`
`nsig2 = 1`)

`sig1`, `sig2`, `itrsig`, `inomde`, `istsig`, `itrds`, `dfsig`, `nsig`: determine trial frequency.

`itrsig` $= 0$: trial frequency taken from `sig1`.

`itrsig` $= 1$: `sig` is found initially from `sig1`, and then, for `jstsig` $= 2, \ldots,$ `istsig`, from previous value `sigp` and `dfsig`.

Meaning of `dfsig` depends on `nsig`:

`nsig` $= 1$: `sig` $=$ `sigp` $+$ `dfsig` nsig = 2: dfsig is increment in frequency (*i.e.*, in $\sqrt{\texttt{sig}}$),

  sig = $(\sqrt{\texttt{sigp}} + \texttt{dfsig})^2$.

nsig = 3: dfsig is increment in 1/(frequency) (*i.e.*, in $1/\sqrt{\texttt{sig}}$),

  sig = $(1/\sqrt{\texttt{sigp}} + \texttt{dfsig})^{-2}$.

(Note that itrsig and nsig have a special meaning when iscan > 1, see below.)

itrsig = 2 or 3: find trial frequency from values on grand summary residing on unit itrds. If itrsig = 2 mode no inomde + jstsig - 1 is taken, and el is reset to the value for this mode. If itrsig = 3 the mode with the value of l input as el and of order inomde + jstsig - 1 is used. Here jstsig is stepped through $1, \ldots,$ istsig.

itrsig = 4 or 5: find trial frequency from values on short summary residing on unit itrds. Otherwise corresponds to itrsig = 2 or 3 above.

itrsig = 6 or 7: find trial frequency from cyclic frequencies (in μHz) from file residing on d/s itrds; the file is in ASCII, each record containing l, n, ν, where ν is the cyclic frequency in μHz. Otherwise itrsig = 6 and 7 corresponds to itrsig = 2 and 3 above.

itrsig = –2 — –5: Find trial frequencies from single-precision files of grand or short summaries, as above for itrsig = 2 — 5.

(Note that istsig has a special meaning when iscan > 1; see below.)

```
(default: itrsig = 0
          sig1 = 10
          sig2 = 0
          dfsig = 0.
          nsig = 1
          istsig = 1
          inomde = 1
          dfsig = 10)
```

iscan: flag for scan in sig. When iscan > 1 step in sig with iscan steps between sig1 and sig2. Step is uniform in squared frequency, frequency or period, for nsig = 1, 2, and 3, respectively. When istsig ≤ 1, sig1 and sig2 are limits in squared dimensionless frequency σ^2; otherwise sig1 and sig2 are the limits in cyclic frequency ν, measured in mHz. When itrsig = 1 check for change of sign in the matching determinant and iterate for eigenfrequency at each change of sign.

```
(default: iscan = 1)
```

eltrw1, eltrw2, sgtrw1, sgtrw2: windows applied to trial l or σ^2. Only takes effect if eltrw1 $\leq$ eltrw2, respectively sgtrw1 $\leq$ sgtrw2.

```
(default: eltrw1 = 0
          eltrw2 = -1
          sgtrw1 = 0
          sgtrw2 = -1)
```

c) Fundamental constants (group cst):

cgrav: Value of gravitational constant (the default is value hardcoded in the programme before 23/3/88)

(default: cgrav = 6.6732×10^{-8})

d) Controls for equations, boundary conditions and integration method (group int):

iplneq: when iplneq = 1 use equations for plane-parallel layer (note that model coefficients should then also correspond to plane-parallel case). Only implemented for non-radial oscillations; attempted calculations with $l = 0$ are skipped, and a diagnostics is printed.

(default: `iplneq = 0`)

`iturpr`: flag for turbulent pressure in equilibrium model

(default: `iturpr = 0`)

`icow`: flag for Cowling approximation.

`icow` = 0: solve full equations.

`icow` = 1: solve equations in Cowling approximation and go back and solve full equations with Cowling result as trial.

`icow` = 2: solve equations in Cowling approximation only. Also sets frequencies corrected by perturbation technique.

`icow` = 3: as `icow` = 2, except that Richardson extrapolation of cyclic frequency is based on uncorrected eigenfrequency.

(default: `icow = 0`)

`alb`: fudge factor in Poisson's equation (the λ in equation 2.3). Should be 1, except when studying gradual transition between Cowling approximation and full case.

(default: `alb = 1.`)

`istsbc`, `fctsbc`: determines surface pressure boundary condition.

`istsbc` = 1: find condition by matching to exponentially decaying solution in an isothermal atmosphere matched to the outermost mesh point [*i.e.*, use conditions (2.17) - (2.20)]. This assumes that the frequency is below the acoustical cut-off frequency at that point. Otherwise a message is printed and the condition for `istsbc` = 0 is used.

`istsbc` = 0: use simple surface condition as given in equations (2.16c) and (2.16d). When `fctsbc` = 0 (the default) this corresponds to $\delta p = 0$.

`fctsbc`: determines condition when `istsbc` = 0: the conditions (2.16c) or (2.16d) with $F_{\rm SBC}$ = `fctsbc`. Note that `fctsbc` = 0 corresponds to using $\delta p = 0$, and `fctsbc` = 1 corresponds to using $p' = 0$. However, all intermediate cases are allowed.

```
(default: istsbc = 1
          fctsbc = 0)
```

`ibotbc`, `fcttbc`: determines bottom boundary condition in truncated model.

`ibotbc` = 0: set relation between y_1 and y_2 at bottom to isolate solution that decreases exponentially towards the interior (*i.e.*, use condition 2.11). This assumes that the bottom is in an evanescent region for the l-value and frequency used. Otherwise the condition corresponding to `ibotbc` = 1 is used.

`ibotbc` = 1: use condition (2.12), with $F_{\rm BC}$ = `fcttbc`.

`ibotbc` = 2: use condition (2.13), *i.e.*, vanishing gradient of displacement.

`ibotbc` = 3: use condition (2.14), *i.e.*, vanishing gradient of relative displacement.

```
(default: ibotbc = 0
          fcttbc = 0)
```

`mdintg`: determines type of integration used (see Sections 3.1 and 3.2).

`mdintg` = 1: use shooting method with centred differences equation (3.11).

`mdintg` = 2: use shooting method with constant coefficient integration on each mesh interval (only implemented for second-order systems, *i.e.*, radial or Cowling approximation; if `mdintg` = 2 and $l > 0$ `icow` = 2 is forced, and a diagnostics is printed).

`mdintg` = 3: use relaxation method. Find frequency by iterating on interior boundary condition for `xfit` = 0, on outer boundary condition for `xfit` = 1, or on matching at an internal point for 0 < `xfit` < 1.

(default: `mdintg = 1`)

`iriche`: if `iriche` = 1, Richardson extrapolation is used to improve the computed frequency (but not the other quantities), by solving initially the equations on every second meshpoint. Note:

trial frequency is decreased by `dsigre`. This option is currently only implemented for `mdintg` = 1 or 3.

(default: `iriche = 0`).

`xfit`: match solution at mesh point $x_{\rm f} = x^{n_{\rm f}} =$ `xfit`. (Note that fitting point may be reset automatically by programme; see Section 3.5). For `xfit` $= -1$ and $l > 0$ match solution at edge of inner evanescent region.

(default: `xfit = 0.5`)

`fcnorm`: for `mdintg` = 3 and `xfit` = 0 or 1, normalize boundary condition by solution at point $x^{(\rm norm)} = x^{n_{\rm norm}}$, where $n_{\rm norm} =$ `fcnorm`*`nn`.

(default: `fcnorm = 0.5`)

`eps`: convergence of `sig` assumed when relative change in `sig` between two iterations is less than `eps`.

(default: depends on internal precision, as set in variable `epsprc` in `common/cprcns/`).

`epssol`: when `mdintg` = 1 or 2 convergence criterion for eigenfunction is assumed to be that relative discontinuity at matching point is less than `epsol`. When `mdintg` = 3 convergence criterion is assumed to be that the mean relative change in the eigenfunction between two iterations is less than `epssol`.

(default: `epssol` = 10^{-5})

`itmax`: maximum number of iterations.

When `itmax` = 0 just integrates once and, when `mdintg` = 1 or 2, tests the continuity of the eigenfunction. This is useful for re-computing eigenfunctions when the eigenfrequency is known.

(default: `itmax = 8`)

`dsigre`: when using Richardson extrapolation, decrease trial `sig` by `dsigre` before iteration on thinned-out mesh

(default: `dsigre = 0`)

`fsig`: in the secant iteration for `sig`, the second value is (1+`fsig`)*(the first trial value)

(default: `fsig = 0.001`)

`dsigmx`: the relative change in `sig` during the iteration is limited to be less than `dsigmx` in absolute value.

(default: `dsigmx = 0.1`)

`irsevn`: Controls scaling of solution in evanescent region (*cf.* Section 3.3).

`irsevn` $= -1$: do not use scaling in evanescent region.

`irsevn` ≥ 1: reset transition point $x_{\rm ev}$ between modified and standard equations (at boundary of evanescent inner region for the trial `sig`) before iterating for each eigenvalue, when iterating for eigenfrequency during scan (*i.e.*, for `iscan` > 1, `itrsig` $= 1$).

`irsevn` = 2: reset fitting point if it is deeper than the evanescent transition (the default is to shift instead the evanescent transition to fitting point).

(default: `irsevn = 2`)

`xmnevn`: Search for evanescent transition is restricted to $x \geq$ `xmnevn`.

(default: `xmnevn = 0`)

`nftmax`, `itsord`: When `nftmax > 1`, attempt changing fitting point `nftmax` times, if iteration does not converge, or, if `itsord = 1`, if the correct order is not obtained. Note that this is most likely to be effective for `mdintg = 3`.

`itsord`: Test on order obtained from trial solution. This is only usable for `itrsig` $= 2 - 7$.

(default: `nftmax = 3`
`itsord = 0`)

e) Controls for output (group `out`):

`istdpr`: unit number for printed output. Default set in block data subprogram `blstio`.
`nout`: when `nout` > 0 print solution at `nout` points
(default: `nout = 50`)
`nprcen`: if `nprcen` > 1 print solution at all the `nprcen` points closest to the centre.
(default: `nprcen = 0`)
`irsord`: Controls setting of order for modes of degree $l = 1$, when the Cowling approximation is not used (*cf.* Section 4.2).
`irsord` between 1 and 10: Increment Scuflaire order between 0 and `irsord` by 1.
`irsord` $= -11$ or 11: Set order according to Lee's scheme. When `irsord` $= -11$, evaluate both values of the order, and write diagnistics if they differ.
(default: `irsord = 0`).
`iekinr`: Determines normalization of energy (*cf.* Section 4.3).
`iekinr` $= 0$: normalize energy with surface vertical displacement.
`iekinr` $= 1$: normalize with total photospheric displacement.
(default: `iekinr = 0`).
`iper`, `ivarf`, `kvarf`, `npvarf`: controls for calculation of the variational frequency.
`iper`: when `iper` $= 1$ calculate variational frequency.
(default: `iper = 0`)
`ivarf` $= 1$: use p-mode formulation (equations (D2) and (D3) of CD82).
`ivarf` $= 2$: use g-mode formulation (equations (D7) and (D8) of CD82).
`ivarf` $= 3$: use nonradial p-mode formulation also for radial modes. (see note in Section 4.4).
(default: `ivarf = 1`)
`kvarf`: numerical differentiation and integration in calculation of variational frequency uses polynomials of degree 2*`kvarf`
(default: `kvarf = 2`)
`npvarf`: for `npvarf` > 0 print integrands and integrals in variational calculation at `npvarf` points.
(default: `npvarf = 0`)
`nfmode`: for `nfmode` $= 1$, 2 or 3 write eigenfunctions to file on unit 4. The output format depends on the value of `nfmode` (see also Section 8.4):
`nfmode = 1`: full set of variables.
`nfmode = 2`: displacement eigenfunctions $y_1(x^n), y_2(x^n)$.
`nfmode = 3`: density-weighted displacement eigenfunctions $\hat{z}_1(x^n), \hat{z}_2(x^n)$ (*cf.* equation 4.3).
(default: `nfmode = 0`)
`irotkr`, `nprtkr`: for `irotkr` $= 1$ calculate rotational kernels. If in addition `nprtkr` > 1 rotational kernel is printed at `nprtkr` points.
(default: `irotkr = 0`
`nprtkr = 50`)
`igm1kr`, `npgmkr`: for `igm1kr` $= 1$ calculate Γ_1 kernels (*cf.* equation 4.7). If in addition `npgmkr` > 1, Γ_1 kernel is printed at `npgmkr` points.
(default: `igm1kr = 0`
`npgmkr = 50`)
`ispcpr`: if `ispcpr` $\neq 0$ special output may be produced by call of user-supplied routine `spcout`.
(default: `ispcpr = 0`)
`icaswn`: If `icaswn` ≥ 0, only output modes to file for which `icase = icaswn`
(default: `icaswn` $= -1$)

`sigwn1`, `sigwn2`, `frqwn1`, `frqwn2`, `iorwn1`, `iorwn2`, `frlwn1`, `frlwn2`: Windows for output of modes to file. Windowing is applied only if first parameter is $\le$ second parameter (*e.g.* `sigwn1` $\le$ `sigwn2`). Defaults are no windowing.
`sigwn1, sigwn2`: Window in σ^2
`frqwn1, frqwn2`: Window in cyclic frequency ν (in μHz)
`iorwn1, iorwn2`: Window in mode order
`frlwn1, frlwn2`: Window in $\nu/(l+1/2)$ (ν in μHz)
(default: `sigwn1 = 0`
`sigwn2 =` -1
`frqwn1 = 0`
`frqwn2 =` -1
`iorwn1 = 0`
`iorwn2 =` -1
`frlwn1 = 0`
`frlwn2 =` -1)

f) Controls for diagnostics (group `dgn`):

`itssol`: when `itssol` $= 1$ test solution with `nrkm` right hand side routine. For `idgtss` $= 1$ additional details about solution etc. are printed.
(default: `itssol = 0`
`idgtss = 0`)
`moddet`, `iprdet`: flags for modifying and printing matching determinant, for `mdintg` $= 1$ or 2.
(default: `moddet = 0`
`iprdet = 0`)
`npout`: when finding eigenfrequency by matching and `npout` > 0 print the separate solutions at `npout` points.
(default: `npout = 0`)
`imstsl`, `imissl`, `imjssl`: Parameters controlling the determination of the matching coefficients in the full case. The equation no. `imissl` is ignored, and the coefficient no. `imjssl` is given a fixed value. If `imstsl` $\neq 1$, `imissl` and `imjssl` are taken as input. Otherwise `imissl` and `imjssl` are determined to minimize the error.
(default: `imstsl = 1`
`imissl = 2`
`imjssl = 2`)
`idgnrk`: determines level of diagnostics in nrkint solution, for `mdintg` $= 3$.
(default: `idgnrk = 0`).

7.3 Comments on the input parameters

The number of parameters may appear overwhelming. They have been introduced to provide a great deal of flexibility in the handling of the input model, the choice of setting up trial frequencies, the execution of the calculation, and the control of the output to file. In practice, standard settings of most parameters are quite adequate; furthermore, template input files can be provided which illustrate the various usages. As a further aid, this section discusses some of the possibilities.

Certain settings of the parameters are inconsistent. I have attempted to catch these cases. When the intended usage is fairly clear, the offending parameters is reset, with a warning message; otherwise an error message is written and the programme proceeds to the next set of input. There is no guarantee that all possible combinations are checked and caught, however.

To select modes on input or output, 'windowing parameters' are typically used. For example, on output modes in a selected frequency range $\nu_1 \leq \nu \leq \nu_2$ can be selected by setting `frqwn1 =` ν_1, `frqwn2 =` ν_2. Windowing is not invoked (fairly obviously) when the lower limit exceeds the upper one, in the present case when `frqwn1` $>$ `frqwn2`.

7.3.1 Notes on model

The default `ifind` $=-1$ ensures reading the first model on unit `imlds` and using it in all the calculations. If the unit number `imlds` is changed in a subsequent read of control input, however, `ifind` must be set ≥ 0 to force reading a new model.

Modifications of the model are controlled by the parameters `xtrnct`, `ntrnsf` and `imdmod`. These modifications may be applied to a model previously read in by setting `ifind =` -2.

Choosing `in` > 1 allows using a coarser model than provided on the file. This may be useful, in saving computing time, for computing low-order modes where high resolution is not required, or for an initial survey of the modes in a model; it is also useful for estimating the effect of truncation errors on the eigenvalues and eigenfunctions. Note that the truncation parameter `ntrnsf` refers to the mesh *after taking every* `in`*-th point.*

7.3.2 Notes on l and trial eigenfrequency

The large number of parameters may make this somewhat complex. However the default values allow simple use, while on the other hand the parameters permit very efficient computation of large sets of modes. The two chief options are

a) To search for modes without any prior information about their location. This would typically be used for general stellar models, where no previously computed set of frequencies is likely to be available.
b) To use a previously computed set of frequencies for trial. This would often be used for solar models which are likely to be quite similar to existing models for which extensive mode sets are available.

a) Searching for modes

To find a mode with a given degree l and near a given trial frequency $\sigma_{\rm tr}$, user only has to provide l in `el` $\sigma_{\rm tr}^2$ in in `sig1`, leaving all other parameters in the group `osc` at their default values. To obtain a sequence of modes with the same l, for which the separation between consecutive modes is approximately known, one uses `itrsig` $= 1$, and sets `istsig` > 1. Then the programme attempts to find `istsig` modes, the first with σ^2 at `sig1`, and the remaining determined from the last obtained eigenvalue and the increment `dfsig`. The three possible ways of specifying the increment, as determined by `nsig`, reflects the possible asymptotic behaviour of the eigenfrequencies

- `nsig` $= 1$: For low-order p modes the frequencies are approximately uniformly spaced in σ^2; here `dfsig` provides the increment between modes in σ^2.
- `nsig` $= 2$: For high-order p modes the spacing is asymptotically uniform in σ; here `dfsig` provides the increment between modes in σ.
- `nsig` $= 3$: For high-order g modes the spacing is asymptotically uniform in $1/\sigma$; here `dfsig` provides the increment between modes in σ^{-1}. Note that in this case the modes are obtained in the order of decreasing frequency and order.

Using `iscan` > 1 permits searching a given region of the spectrum for eigenmodes, between $\sigma^2 =$ `sig1` and $\sigma^2 =$ `sig2`. The step is uniform in σ^2, σ or σ^{-1} depending on `nsig`, as discussed above. When `itrsig` $\neq 1$ the programme only prints a table of the matching determinant Δ (defined by equation (3.5), (3.9), (3.13) or (3.15), depending on the integration parameters), as a function of σ^2. This gives an initial idea about the spectrum. For `itrsig` $= 1$ the programme in addition looks for changes of sign in Δ, and at each change of sign (except those associated with singularities for `mdintg` $= 3$; *cf.* Section 3.2) it attempts to iterate for the eigenvalue. Thus automatic determination of all modes in a given region of the spectrum is possible. To ensure that all modes are found with reasonable certainty the number `iscan` of steps should probably be at least about four times the number of modes expected in the region considered, and `nsig` should of course be chosen appropriately for the kind of modes expected. It might also be pointed out that two very close eigenvalues (near an avoided crossing, say) may manifest themselves as a minimum in $|\Delta(\sigma^2)|$, without changes of sign; in this case the search may be repeated with smaller step near the minimum (this procedure could clearly be automated, but that has so far not been done).

Instead of considering a single value of l given in `el`, the programme may be asked to step in l. This is accomplished by setting `nsel` > 1; `nsel` is the number of l-values considered, `els1` is the initial value of l and `dels` is the step in l. This may be used in connection with all the options mentioned above:

- To follow a single mode, or step through σ^2 with `istsig` > 1, `itrsig` $= 1$, the initial trial σ^2 must be provided in `sig1`. For each step in l, the trial σ^2 is incremented as determined by `dfsig1` and `nsig1`, in the same way as described for `dfsig`, `nsig` above.
- To scan in σ^2 with `iscan` > 1, the initial range in σ^2 must be provided in `sig1`, `sig2`. For each step in l, the end points of the range are incremented as determined by `dfsig1`, `nsig1` and `dfsig2`, `nsig2`, as above.

```
osc:
el, nsel, els1, dels, dfsig1, dfsig2, nsig1, nsig2
,8,  30  ,10,   10.,    10.,    1,    1     @
itrsig, sig1, istsig, inomde, itrds,
1,    30.  ,     ,      ,      ,      @
dfsig, nsig, iscan, sig2,
,   1   ,10,  34    @
eltrw1, eltrw2, sgtrw1, sgtrw2
,        ,       ,        ,    @
```

Figure 1. Input block to scan for f mode.

Example: To compute the f mode for degrees $l = 30, 40, \ldots, 100$ by scanning in σ^2, the `osc` group of input shown in Figure 1 may be used. This will scan in σ^2 between $\sigma^2 = l$ and $\sigma^2 = l + 4$ for each value of l, and hence is likely to find the f mode, with σ^2 slightly exceeding l.

b) Using previously computed frequencies as trials

The options of having `itrsig` ≥ 2 and `istsig` ≥ 1 allow easy computation of modes of a model, given results on the same modes in a slightly different model. The results on the modes can be in the form of binary files containing a grand or a short summary (see Sections 8.2 and 8.3 below), or an ASCII file containing degree, order and cyclic frequency (in μHz). (The possibility of specifying itrsig ≤ -2 was introduced to allow use of older files of summaries in single precision; it is unlikely to be of general usefulness.)

There are two different options:

- When itrsig is even, the modes are located according to their position on the file, by specifying the range inomde, inomde $+ 1, \dots,$ inomde + istsig $- 1$ of mode numbers in the file; both σ^2 and l are taken from the values on the file.
- When itrsig is odd the programme searches for modes of a specified degree and with orders inomde, inomde$+1, \dots,$ inomde$+$ istsig-1. The degree is either provided in el or, by setting nsel > 1, by stepping through a sequence of degrees.

In both cases the search can be restricted to given ranges eltrw1, eltrw2 in degree and/or (sgtrw1, sgtrw2) in σ^2 by having eltrw1 $\leq$ eltrw2 and/or sgtrw1 $\leq$ sgtrw2.

```
osc:
el, nsel, els1, dels, dfsig1, dfsig2, nsig1, nsig2
,21,  5  ,5,   0.,    0.,   1,    1     @
itrsig, sig1, istsig, inomde, itrds,
3,       ,   9 ,    2 ,      ,       @
dfsig, nsig, iscan, sig2,
,      ,    ,    ,    @
eltrw1, eltrw2, sgtrw1, sgtrw2
,        ,   100 ,   1.e6    ,    @
```

Figure 2. Input block to computes modes of given order, based on trial frequencies on file.

The possibility of combining itrsig odd with nsel > 1 allows the computation of modes of a given order for several values of l. Figure 2 illustrates the osc input group for computing modes of order $2, 4, \dots 10$, for $l = 0, 5, \dots, 100$, and restricting σ^2 to exceed 100.

The choice of itrsig $= 6$ or 7 allows convenient computation of frequencies on the basis of a set of observed modes. A possible input format is illustrated in Figure 3. Note that the file may contain a header, indicated by the character # in the first column. The data are read in using free format. Also, the file may contain additional columns beyond the degree, order and frequency, in the case illustrated the standard error. These columns are ignored.

7.3.3 Remaining parameters

The meaning of most of the parameters controlling the equations and boundary conditions should be reasonably clear on the basis of the discussion in Section 2. The case of a plane-parallel layer (iplneq $= 1$) is discussed in Section A.2; iturpr is used to control the treatment of turbulent pressure in a realistic model of the solar atmosphere, as discussed in Section A.3.

The parameters controlling the integration are essentially discussed in Section 3, with the exception of the test on the discontinuity of the eigenfunction. This is based on the absolute value of the determinant in equation (3.5) or (3.9), normalized by the product of the norms of the columns; thus in the case of equation (3.5) we introduce

$$\tilde{\Delta} \equiv \frac{|\Delta|}{[(y_1^{(\mathrm{i})}(x_\mathrm{f})^2 + y_2^{(\mathrm{i})}(x_\mathrm{f})^2)(y_1^{(\mathrm{o})}(x_\mathrm{f})^2 + y_2^{(\mathrm{o})}(x_\mathrm{f})^2)]^{1/2}} , \tag{7.1}$$

```
# Data from Libbrecht, K.G., Woodard, M.F., and Kaufman, J.M.,
# ApJS, vol.  74, p.  1129 (1990).
#
#
# Mode Frequency Table (+/- gives 1-sigma random errors)
#    l     n   nu(uHz)    +/-  source
#    ---   --- --------- ------- ------
0    12  1823.600   0.600    2
0    13  1957.300   0.400    2
0    14  2093.500   0.200    2
0    15  2228.600   0.100    2
0    16  2362.500   0.100    2
0    17  2496.600   0.300    2
0    18  2629.600   0.300    2
.
.
.
.
```

Figure 3. Possible structure of input file for the cases `itrsig` = 6 and 7.

and the condition for continuity of the eigenfunction is that $\tilde{\Delta} \le$ `epssol`.

Similarly the meaning of the parameters determining the output should be clear or may be inferred from Section 4 (see also Section 8 below describing the results of the computation). Finally, the parameters controlling the diagnostics are unlikely to be of interest for general users.

7.4 The user-supplied routines `modmod` and `spcout`

If `imdmod` $\neq 0$ `modmod` is called after the equilibrium model has been read (and, if applicable, after taking every `in`-th point), as

```
call modmod(x, aa, data, nn, iaa, imdmod)
```

where `iaa` is the first dimension of `aa`, and is set by the programme calling `modmod`; the remaining quantities are defined above. It has no further effect on the calculations, but may be used by the user to carry out modifications to the model. Similarly, if `ispcpr` $\neq 0$ the routine `spcout` is called at the end of the calculation for each mode, as

```
call spcout(x, y, aa, data, nn, iy, iaa, ispcpr)
```

to allow the user to produce output in addition to that otherwise produced by the programme; here `iy` is the first dimension of `y` and is set by the programme calling `spcout`. When `imdmod` = 0 (or `ispcpr` = 0), as is the default, the routines are not called.

The standard set-up of the code supplies dummy subroutines, in the file containing the main programme `main`. These must obviously be replaced by the user for these options to take effect.

8 Output from the programme

This section contains a description of the principal output produced by the programme. Section 8.1 describes what may somewhat archaically be referred to as printed output, for each mode. The remaining sections gives the format of the output, likely to be of most use, produced on disk files. The unit numbers of the the output files are currently hardcoded into variables in the programme. The relevant unit numbers, with the (hardcoded) defaults and description of the output, are:

`idsgsm` (default 11): Grand summary file
(Section 8.2; conventional name `agsm.<model descriptor>`).
`idsssm` (default 15): Short summary file
(Section 8.3; conventional name `assm.<model descriptor>`).
`idsefn` (default 4): Eigenfunction file
(Section 8.4; conventional name `amde.<model descriptor>[.z]`).
`idsrkr` (default 12): Rotational splitting kernel file (Section 8.5)
`idsgkr` (default 13): Γ_1 kernel file (Section 8.6)
`idslog` (default 20): Log file of error and warning messages from eigenfrequency iteration. If `idslog` is not defined in input file, the default name `adipls-status.log` is used (see Section 8.7).

8.1 Printed output

This is output on unit `istdpr`, which may be modified in the input file (group `out`). If `istdpr` $\neq$ `istdou`, where `istdou` is the unit number for the standard output, a summary of the calculation, including error and warning messages, is also output to `istdou`. Note that `istdou` is normally 6. It is defined in `block data` subprogramme `blstio`.

After some diagnostics relating to the integration follows output related to the iteration for (or scan in) σ^2. For each iteration is given the iteration number, the current value of σ^2, and the corresponding values `ddsig` and `ddsol` of the matching determinant Δ and the normalized determinant $\tilde{\Delta}$ (or, when iterating with `mdintg` = 3, the mean change in the solution since the previous value of σ^2). For `mdintg` = 2 an estimate of the mode order, defined by equation (4.1), is also given. After the iteration has converged the final value of σ^2 is printed, as well as $\tilde{\sigma}^2$ (*cf.* equation 4.1) and the period calculated from σ^2. If the equations are solved in the Cowling approximation the correction $\delta\sigma^2$ and the corrected value σ_c^2 obtained from Cowling's perturbation technique are also printed.

Then follows the maximum absolute value $z_{1,\rm max}$ of the energy-related eigenfunction $z_1(x)$ (*cf.* equation 4.3) and the location $\hat{x}_{max}$ of the maximum. After this comes the printout of the eigenfunction, in the form of

$$n,\ x^n,\ y_1(x^n),\ y_2(x^n),\ y_3(x^n),\ y_4(x^n),\quad \hat{z}_1(x^n),\ \hat{z}_2(x^n),$$

for non-radial oscillations, or

$$n,\ x^n,\ y_1(x^n),\ y_2(x^n),\ \hat{z}_1(x^n),$$

for radial oscillations. Here $\hat{z}_1$ and $\hat{z}_2$ are as defined in equations (4.3) and (4.5).

The labelling of the mode is given next, in the form

```
This is a pn(l = l) mode
This is a f(l = l) mode
This is a g|n|(l = l) mode
```

for the order $n > 0$, $n = 0$ or $n < 0$ respectively. After this follows the maximum absolute value $y_{1,\rm max}$ of $y_1(x)$ and the location $x_{\rm max}$ of the maximum, and the dimensionless energy E of pulsation (*cf.* equation 4.2).

If `iper` has been set to 1 there next follows output from the calculation of the variational frequency and the corresponding frequency. Here `sigv` gives the value of σ^2 obtained from the variational expression; in addition the period Π_E found from the value of σ^2 obtained as an eigenvalue is repeated, and the period Π_V and cyclic frequency ν_V found from the variational σ^2 are printed.

If the rotational kernel is calculated (*cf.* equation 4.6), the value of β_{nl} is printed and, if `nprtkr` > 0, the kernel is printed, as

$$n, \ x^n, \ K_{nl}(x^n) \ .$$

8.2 The grand summary

A fairly complete summary of the calculation is written to unit `idsgsm`, without format; this only takes place if the eigenvalue iteration has converged. The summary consists of 38 real variables `cs(1:38)` and 8 integer variables `ics(1:8)`. These are defined as follows (`typewrite-type names` refer to the list of input parameters, *cf.* Section 7.2.3):

- `cs(1)`: `xmod`
- `cs(2:8)`: D_1 - D_7:
  - `cs(2)`: M
  - `cs(3)`: R
  - `cs(4)`: p_c
  - `cs(5)`: ρ_c
  - `cs(6)`: $-(1/\Gamma_1 p)(\mathrm{d}^2 p/\mathrm{d}x^2)$ at centre
  - `cs(7)`: $-(1/\rho)(\mathrm{d}^2\rho/\mathrm{d}x^2)$ at centre
  - `cs(8)`: μ
- `cs(9)`: unused
- `cs(10)`: $A_2(x_\mathrm{s})$
- `cs(11)`: $A_5(x_\mathrm{s})$
- `cs(12)`: x_1
- `cs(13)`: unused
- `cs(14)`: x_f
- `cs(15:16)`: `fctsbc`, `fcttbc`
- `cs(17)`: λ (fudge factor in Poisson's equation)
- `cs(18)`: l (degree)
- `cs(19)`: n (order)
- `cs(20)`: σ^2
- `cs(21)`: σ_c^2
- `cs(22:23)`: $y_{1,\mathrm{max}}$, x_max
- `cs(24)`: E (dimensionless energy)
- `cs(25:27)`: Π_E, Π_V, ν_V,
- `cs(28:29)`: `ddsig`, `ddsol`
- `cs(30:33)`: $y_1(x_\mathrm{s}) - y_4(x_\mathrm{s})$
- `cs(34:35)`: $z_{1,\mathrm{max}}$, $\hat{x}_\mathrm{max}$
- `cs(36)`: β_{nl}
- `cs(37)`: ν_Ri, *i.e.*, cyclic frequency from Richardson extrapolation, (in mHz).
- `cs(38)`: unused

- `ics(1)`: `in`

`ics(2)`: `nn`
`ics(3)`: `mdintg`
`ics(4)`: `ivarf`
`ics(5)`: `icase`
`ics(6)`: `iorign`
`ics(7)`: `iekinr`
`ics(8)`: unused.

Notes:

- Depending on the parameters of the calculation, some variables may not be set. All unset variables are initialized to 0. In particular `cs(9)`, `cs(13)`, `cs(38)` and `ics(8)` are currently unused.
- `ddsig` and `ddsol` [*i.e.*, `cs(28)` and `cs(29)`] are set to the values of Δ and $\tilde{\Delta}$ in the last iteration, and thus provide a measure of the extent to which the iteration has converged.
- When Cowling approximation is used Π_E (in `cs(25)` is set to the period corresponding to the *uncorrected* eigenfrequency.
- `icase` is a compressed case number for the calculation. It is defined as

  ```
  icase = icow1 + 10 iper + 1000 ispec + 10 000 istsbc
      + 100 000 iplneq + 1 000 000 iturpr
  ```

  Here `icow1` is defined as follows:
  - `icow1` = is 0 when the full set of equations is used.
  - `icow1` = 1 when the Cowling approximation is used, and the Richardson extrapolated frequency is based on the corrected eigenfrequency σ_c.
  - `icow1` = 2 when the Cowling approximation is used, and the Richardson extrapolated frequency is based on the uncorrected eigenfrequency (*i.e.*, computed with the input parameter `icow` = 3).

  Furthermore `ispec` = 1 if $\lambda \neq 1$ or `fctsbc` $\neq 1$, and the remaining parameters are as defined in Section 7.2.3.
- `iorign` is always set to 3 in the pulsation programme (it may have other values for grand summaries set on the basis of other, more restricted sources of information).

`ics` is stored in the array `cs(1:50)`, by the following equivalencing

```
equivalence(ics(1), cs(39))
```

Thus if reals and integers have the same length only the first 46 positions in `cs` are used, whereas (as is typically the case) if 8-byte real variables and 4-byte integers are used, only the first 42 positions of `cs` are used*. In the programme `cs` is stored in `common/csumma/`.

The summary is output in a single record for each mode, by the statement

```
write(idsgsm) (cs(i), i=1,50)
```

Various programmes are available to scan or manipulate the contents of the grand summary (see *Notes on using the solar models and adiabatic pulsations package*). In particular, the programme `scan-agsm.d` scans the grand summary, whereas `set-obs.d` outputs a file giving degree, order, cyclic frequency and possibly mode energy.

* This is a historical consequence of a previous possibility of storing a model name in the last part of `cs`; in future, an extension or generalization of the storage might be contemplated.

8.3 The short summary

To save disk space, a condensed summary has been introduced. This contains the most essential information about the modes, without the details that are rarely used. Thus it is adequate for most purposes. In addition data on the model are given as special records, rather than being repeated in each record.

The data are stored in the real array `ss(1:5)` and the integer array `iss(1:2)`.

The type of the record is flagged by `ss(1)`:

Model record : This is flagged by `ss(1)` < 0. `ss(2)` is set to `xmod`. The remaining variables are defined by

`ss(3)`: M
`ss(4)`: R
`ss(5)`: p_c
`ss(6)`: ρ_c

Oscillation record: This is flagged by `ss(1)` ≥ 0. Here

`ss(1)`: l (degree)
`ss(2)`: n (order)
`ss(3)`: σ^2
`ss(4)`: E (dimensionless energy)
`ss(5)`: ν_V (in mHz)

`iss(1)`: `icase`
`iss(2)`: `iorign`

Note that in the Cowling approximation σ^2 is taken to be the corrected value σ_c^2 obtained with the Cowling perturbation expression. Also `ss(5)` is the *variational* frequency if this has been calculated; otherwise the frequency obtained as an eigenvalue is used. `iss(1:2)` are stored in `ss(1:7)`, by the following equivalence statement

```
equivalence (ss(6), iss(1))
```

The summary is written in a single record, with the statement

```
write(idsssm) (ss(i), i=1,7)
```

8.4 Output of eigenfunction to file

If `nfmode` = 1, 2 or 3 and the eigenfunction iteration has converged the eigenfunctions are written on unit 4, as a single record, in binary form. The format depends on the value of `nfmode`:

a) `nfmode = 1`, full set of eigenfunctions.

Here the output is done with the statement

```
write(idsefn) (cs(i), i=1,50), nnw, (x(n), (y(i,n), i=1,6), n=nw1,nn)
```

Here `nnw = nn - nw1 + 1`, `y(1:4,n)` contains the eigenfunctions $y_1(x^n)$ – $y_4(x^n)$ (*cf.* Section 6), and

$$\mathtt{y(5,n)} = \hat{z}_1(x^n), \quad \mathtt{y(6,n)} = \hat{z}_2(x^n)$$

[*cf.* equations (4.3) and (4.5)]. An eigenfunction file of this format conventionally has name `amde.<model descriptor>`.

b) `nfmode = 2`, displacement eigenfunctions.

Here the first record contains the number of mesh points and the mesh, written as

```
    write(idsefn) nnw, (x(n), n=nw1,nn)
```

and the subsequent records contain the mode data, in the form

```
    write(idsefn) (cs(i), i=1,50), ((y(i,n), i=1,2), n=nw1,nn)
```

c) `nfmode = 3`, density-weighted displacement eigenfunctions.

Here the first record contains the number of mesh points and the mesh, written as

```
    write(idsefn) nnw, (x(n), n=nw1,nn)
```

and the subsequent records contain the mode data, in the form

```
    write(idsefn) (cs(i), i=1,50),((z(i,n), i=1,2), n=nw1,nn)
```

where

$$\mathtt{z(1,n)} = \hat{z}_1(x^n), \quad \mathtt{z(2,n)} = \hat{z}_2(x^n)$$

[*cf.* equations (4.3) and (4.5)]. An eigenfunction file of this format conventionally has name `amde.<model descriptor>.z`.

Thus in all cases the eigenfunction record contains the grand summary (*cf.* Section 8.3). Format a) is clearly the most space-consuming. Format c) produces eigenfunctions in a form suitable for setting up kernels. In particular, to compute rotational kernels no further information about the equilibrium model is required. This is therefore most likely the format of choice, unless the full set is explicitly needed.

8.5 Output of rotational kernel to file (*cf.* equation 4.7)

If `irotkr` = 1 the rotational kernel is written on unit 12, as a single record, without format. This is done with the statement

```
    write(idsrkr) (cs(i), i=1,50),nnw, (x(n), akr(n), n=nw1,nn)
```

Here

$$\mathtt{akr(n)} = K_{nl}(x^n),$$

Thus the record contains the grand summary (*cf.* Section 8.3).

8.6 Output of Γ_1 kernel to file (*cf.* equation 4.8)

If `igm1kr = 1` the Γ_1 kernel is written on unit 13, as a single record, without format. This is done with the statement

```
    write(idsgkr) (cs(i), i=1,50), nnw, (x(n), akrgm1(n), n=nw1,nn)
```

Here

$$\mathtt{akrgm1(n)} = K_{nl}^{(\Gamma_1)}(x^n),$$

Thus the record contains the grand summary (*cf.* Section 8.3).

8.7 Iteration status log

A log of problems with the eigenfrequency calculation is output to unit `idslog`; if no file name is provided, the default is `adipls-status.log`. This may contain both error and warning messages:

— An error message is printed if
  - The iteration failed to converge, even after possible attempts of adjusting x_f (*cf.* Section 3.5).
  - Significantly different eigenfunctions were found in Richardson-extrapolation calculation (*cf.* Section 3.4).

  Note that in cases of errors, no output is made to the files with grand summaries, eigenfunctions, *etc.*

— A warning message is printed if
  - The location of x_f had to be changed to obtain convergence or the correct order.
  - Different mode orders were found in Richardson-extrapolation calculation, but the eigenfunctions were deemed to be sufficiently similar.

  In cases of warnings, normal output is made to the files with grand summaries, eigenfunctions, *etc.* Indeed, in these cases one can generally assume that the calculation has been successful.

9 The main programme

The calculation is controlled by the call of the subroutine `adipls`. A small main programme is needed to set up storage for the calculation. This has the form

```
      program runadi
c
c  main programme for adiabatic pulsations
c
c  quantities set in parameter statement:
c  nnmax:  maximum number of mesh points in integration
c
c  Note:  as presently set up, nnmax is set also in s/r nrkint and
c  eigin4
c
c  Double precision version.
c  ++++++++++++++++++++++++
c
c  Dated:  10/3/90
c
       implicit double precision (a-h, o-z)
       parameter (nnmax = 10000)
       parameter (nnmax1 = nnmax+1, nnmax2 = nnmax+10)
       common/rhsdat/ dt1(20), aa(6,nnmax) /xarr/ x(nnmax)
     *  /xarr1/ x1(nnmax1) /xarr2/ x2(nnmax1)
     *  /worksp/ aa1(9,nnmax)   /yyyyyy/ y(8,nnmax)
     *  /yyyyri/ yri(4,nnmax)
     *  /sysord/ sysyso(4)
     *  /work/ wwnrk(20,nnmax2)
       common/wrkleq/ wwwlll(1500)
       common/cderst/ derc(6,nnmax)  /cintst/ aintc(6,nnmax)
c
c  common defining standard input and output
c
```

```
      common/cstdio/ istdin, istdou, istdpr
c
      write(istdou,100) nnmax
c
      call adipls
      stop
  100 format(//61('*')//
     * ' In this version, the maximum number of mesh points is', i5//
     * 61('*'))
      end
```

Here `nnmax` is the maximum number of mesh points *used in the calculation*, and `ii` is the order of the system. Furthermore `common /worksp/` should be large enough to contain the array `aain(6,nnmodl)`, where `nnmodl` is the total number of mesh points in the equilibrium model read in (if `in` is not equal to 1, `nnmodl` is larger than `nn`).

The work space set in `common/work/` is only needed for integration with the relaxation technique, *i.e.*, for `mdintg` = 3. If `mdintg` = 1 or 2 the line setting up this work area may be removed.

References

Baker, N. H., Moore, D. W. & Spiegel, E. A., 1971. [Aperiodic behaviour of a non-linear oscillator]. *Q. J. Mech. Appl. Math.*, **24**, 391 – 422.

Christensen-Dalsgaard, J., 1980. [On adiabatic non-radial oscillations with moderate or large l]. *Mon. Not. R. astr. Soc.*, **190**, 765 – 791.

Christensen-Dalsgaard, J., 1981. [The effect of non-adiabaticity on avoided crossings of non-radial stellar oscillations]. *Mon. Not. R. astr. Soc.*, **194**, 229 – 250.

Christensen-Dalsgaard, J., 1982. [On solar models and their periods of oscillation]. *Mon. Not. R. astr. Soc.*, **199**, 735 – 761.

Christensen-Dalsgaard, J. & Frandsen, S., 1983. [Radiative transfer and solar oscillations]. *Solar Phys.*, **82**, 165 – 204.

Christensen-Dalsgaard, J. & Mullan, D. J., 1994. [Accurate frequencies of polytropic models]. *Mon. Not. R. astr. Soc.*, **270**, 921 – 935.

Christensen-Dalsgaard, J., Dilke, F. W. W. & Gough, D. O., 1974. [The stability of a solar model to non-radial oscillations]. *Mon. Not. R. astr. Soc.*, **169**, 429 – 445.

Cowling, T. G., 1941. [The non-radial oscillations of polytropic stars]. *Mon. Not. R. astr. Soc.*, **101**, 367 – 375.

Gabriel, M. & Noels, A., 1976. [Stability of a $30\,M_\odot$ star towards g^+ modes of high spherical harmonic values]. *Astr. Astrophys.*, **53**, 149 – 157.

Gabriel, M. & Scuflaire, R., 1979. [Properties of non-radial stellar oscillations]. *Acta Astron.*, **29**, 135 – 149.

Lee, U., 1985. [Stability of the Delta Scuti stars against nonradial oscillations with low degree l]. *Publ. Astron. Soc. Japan*, **37**, 279 – 291.

Mihalas, B. W. & Toomre, J., 1981. [Internal gravity waves in the solar atmosphere. I. Adiabatic waves in the chromosphere]. *Astrophys. J.*, **249**, 349 – 371.

Scuflaire, R., 1974. [The non radial oscillations of condensed polytropes]. *Astr. Astrophys.*, **36**, 107 – 111.

Rosenthal, C. S., Christensen-Dalsgaard, J., Houdek, G, Monteiro, M.J.P.F.G., Nordlund, Å. & Trampedach, R., 1995. [Seismology of the solar surface regions]. In: *Proc. Fourth SOHO Workshop: Helioseismology*, eds Hoeksema, J. T., Domingo, V., Fleck, B & Battrick, B., ESA SP-376, vol. 2, ESTEC, Noordwijk, p. 459 – 464.

Rosenthal, C. S., Christensen-Dalsgaard, J. Nordlund, Å., Stein, R. F. & Trampedach, R., 1997. [Convective contributions to the frequencies of solar oscillations]. *Astron. Astrophys.*, in preparation.

Shibahashi, H. & Osaki, Y., 1981. [Theoretical eigenfrequencies of solar oscillations of low harmonic degree ℓ in five-minute range]. *Publ. Astron. Soc. Japan*, **33**, 713 – 719.

Unno, W., Osaki, Y., Ando, H. & Shibahashi, H., 1989. *Nonradial Oscillations of Stars* (2. ed.), University of Tokyo Press.

Appendix A. Equations

In this appendix the equations of non-radial oscillations are given in the form they are solved in the programme. Section A.1 presents the equations in the standard case of oscillations of a spherical star with no turbulent pressure, Section A.2 discusses the implementation used for oscillations of a plane-parallel layer, and Section A.3 discusses the ways turbulent pressure may be treated (or neglected).

A.1 Equations in the standard case

For non-radial oscillations the equations satisfied by the y_i are

$$x\frac{\mathrm{d}y_1}{\mathrm{d}x} = (V_g - 2)y_1 + \left(1 - \frac{V_g}{\eta}\right) y_2 - V_g y_3 \ , \qquad (A.1)$$

$$x\frac{\mathrm{d}y_2}{\mathrm{d}x} = [l(l+1) - \eta A]y_1 + (A-1)y_2 + \eta A y_3 \ , \qquad (A.2)$$

$$x\frac{\mathrm{d}y_3}{\mathrm{d}x} = y_3 + y_4 \ , \qquad (A.3)$$

and

$$x\frac{\mathrm{d}y_4}{\mathrm{d}x} = -\lambda A U y_1 - \lambda U \frac{V_g}{\eta} y_2 \qquad (A.4)$$
$$+[l(l+1) + U(A-2) + (1-\lambda)UV_g]y_3 + 2(1-U)y_4 \ .$$

Here $\eta = l(l+1)g/(\omega^2 r)$, and the notation is otherwise as defined in equation (5.1). Note that the modified form of Poisson's equation, equation (2.3), has been used. In the Cowling approximation the terms in y_3 are neglected in equations (A.1) and (A.2), and equations (A.3) and (A.4) are not used.

For radial oscillations the equations are

$$x\frac{\mathrm{d}y_1}{\mathrm{d}x} = (V_g - 2)y_1 - V_g \frac{\sigma^2 x^2}{q} y_2 \ , \qquad (A.5)$$

and

$$x\frac{\mathrm{d}y_2}{\mathrm{d}x} = \left[x - \frac{q}{\sigma^2 x^2}(A - \lambda U)\right] y_1 + A y_2 \ . \qquad (A.6)$$

A.2 Equilibrium variables and oscillation equations in plane-parallel case

The option has been built into the programme to study *nonradial* oscillations of a plane-parallel envelope model with constant gravity. This option is invoked by setting the parameter `iplneq` to 1. The oscillations are treated in the Cowling approximation. For simplicity the model and the oscillations are still put on dimensionless form in terms of a total mass M and a radius R, which, however, clearly has no specific physical meaning (except that the results may be taken as approximations to envelope oscillations of a star with the given mass and radius). The gravity is then given by $g = GM/R^2$. The horizontal wave number k_h is parameterized in terms of the "degree" l, defined such that

$$k_h = \frac{l}{R} \ . \qquad (A.7)$$

The equilibrium model is still given by x and $A_1 - A_5$. Now, however, x is a measure of position, in units of R, decreasing towards the interior of the model; the point $x = 1$ may still conveniently be chosen at or close to the surface of the model. The variables $A_1 - A_5$ are defined as

$$\begin{aligned} A_1 &\equiv 1 \, , \\ A_2 &= V_g \equiv \frac{g\rho R}{\Gamma_1 p} \, , \\ A_3 &\equiv \Gamma_1 \, , \\ A_4 &= A \equiv -V_g - \frac{\mathrm{d}\ln\rho}{\mathrm{d}x} \, , \\ A_5 &= U \equiv \frac{4\pi\rho R^3}{M} \, . \end{aligned} \tag{A.8}$$

The oscillation equations are now

$$\frac{\mathrm{d}y_1}{\mathrm{d}x} = V_g y_1 + \left(1 - \frac{V_g}{\eta}\right) y_2 \, , \tag{A.9}$$

and

$$\frac{\mathrm{d}y_2}{\mathrm{d}x} = (l^2 - \eta A) y_1 + A y_2 \, , \tag{A.10}$$

where $\eta = l^2/\sigma^2$. Notice that A_1 and A_5 are not needed in the oscillation equations. However, they should still be defined as in equations (A.8), as they are used elsewhere in the calculations in the programme, *e.g.* in the evaluation of the energy in equation (4.3).

The calculation of radial oscillations in a plane-parallel envelope has not been implemented.

A.3 The treatment of turbulent pressure

There is no doubt that turbulence contributes significantly to the hydrostatic balance in the outermost layers of stars with convective envelopes. This effect is often described in terms of a turbulent pressure p_{turb}; in solar models estimates of p_{turb} indicate that it may be as high as $10 - 15$ per cent of the total pressure near the top of the convection zone. Even so, turbulent pressure is often neglected in the calculation of equilibrium models.

If turbulent pressure is included in the equilibrium model, the oscillation calculations require an assumption about the response of p_{turb} to the pulsations. This is sometimes stated by saying that "the perturbation to turbulent pressure is neglected". However, the effects on the frequencies might depend sensitively on the way in which the p_{turb}-perturbation is neglected, often determined by the precise form of the oscillation equations used for the numerical solution.

This programme allows several options for the treatment of turbulent pressure:

i) Neglect of the Eulerian perturbation in the turbulent body force.
ii) Neglect of the Eulerian perturbation in the turbulent pressure.
iii) Neglect of the Lagrangian perturbation in the turbulent pressure.

In all three cases, the equilibrium model, defined by the $\{A_i\}$, must be set up appropriately. In addition, the first two cases must also be flagged by the input parameter `iturpr`. The detailed use of these options is discussed below.

The first option, i), is the original option in the code; it was introduced largely because of computational convenience, but follows the treatment used by Mihalas & Toomre (1981). It was argued by Christensen-Dalsgaard & Frandsen (1983) that neglect of the *Lagrangian* perturbation of the turbulent body force might be somewhat more reasonable, although in no ways satisfactory. This has not been implemented; however, option iii) shares some of the same advantages. Option ii) is retained in the code to allow tests of the sensitivity of the computed frequencies to different assumptions.

The frequency effects of turbulent pressure were discussed in more detail by Rosenthal *et al.* (1995) who found that the effect on the equilibrium structure of turbulent pressure, and the perturbation in turbulent pressure, is of the same order of magnitude as other uncertain aspects of the mode physics, including nonadiabaticity, and also of roughly the magnitude and shape of the current dominant component of the difference between observed frequencies and frequencies of adiabatic oscillation for normal solar models. Thus, although turbulent pressure affects a region of the Sun suffering from other uncertainties in the treatment of the oscillations, the way it is included or ignored is of considerable interest.

The implementation of the different options has not been tested with great care yet, and hence some caution is required when using them. Also, it should be noticed that computation of variational frequencies has not been consistently corrected for turbulent pressure. As discussed below, this should have no effect for option iii) but is likely to influence options i) and ii).

A.3.1 Neglect of the Eulerian perturbation in the turbulent body force (`iturpr = 1`*)*

We write the equations of motion as

$$\rho \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\nabla p + \rho \mathbf{g} + \rho \mathbf{f}_{\rm turb} \; , \tag{A.11}$$

where $\mathbf{g} = \nabla\Phi$ is the gravitational acceleration, Φ being the gravitational potential, and $\mathbf{f}_{\rm turb}$ is the turbulent body force; furthermore p is the combined gas and radiation pressure. Thus the equation of hydrostatic support, when expressed in terms of p, is

$$\frac{\mathrm{d}p}{\mathrm{d}r} = -\tilde{g}\rho \; . \tag{A.12}$$

where $\tilde{g} = g - f_{\rm turb}$, where $-g$ and $f_{\rm turb}$ are the radial components of the equilibrium gravity and turbulent force. Thus here and in the following

$$g = \frac{Gm}{r^2} \; . \tag{A.13}$$

In the oscillation calculation the assumption is now that the Eulerian perturbation of $\mathbf{f}_{\rm turb}$ can be neglected. Hence the Eulerian perturbation of equation (A.11) may be written

$$\rho \frac{\partial^2 \delta\mathbf{r}}{\partial t^2} = -\nabla p' - \rho' \tilde{g} \mathbf{a}_r + \rho \nabla \Phi' \; . \tag{A.14}$$

Evidently, the condition of adiabaticity is expressed as a relation between the Lagrangian perturbations in p and ρ:

$$\frac{\delta p}{p} = \Gamma_1 \frac{\delta\rho}{\rho} \; , \tag{A.15}$$

where Γ_1 is the (usual) thermodynamic adiabatic exponent.

To take the distinction between g and $\tilde{g}$ into account, we modify the definitions of the equilibrium variables A_1, A_2 and A_4. Thus, instead of equations (5.1), we use

$$
\begin{aligned}
A_1 &\equiv \frac{R^3}{M}\frac{\tilde{g}}{r} \equiv \frac{\tilde{q}}{x^3} \;, \qquad \text{which defines} \quad \tilde{q} = \frac{\tilde{g}}{g} q \;, \\
A_2 &= \tilde{V}_g \equiv -\frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} \equiv \frac{r\rho\tilde{g}}{\Gamma_1 p} \;, \\
A_4 &= \frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} - \frac{\mathrm{d}\ln \rho}{\mathrm{d}\ln r} \;,
\end{aligned}
\tag{A.16}
$$

together with A_3 and A_5 which are unchanged. We also introduce

$$
V_g = \frac{r\rho g}{\Gamma_1 p} \;, \qquad A = -V_g - \frac{\mathrm{d}\ln\rho}{\mathrm{d}\ln r} \;. \tag{A.17}
$$

Clearly in the absence of turbulent pressure these variables reduce to the variables defined in Section 5.

Note that with these definitions the *structure* of the model file is unchanged. In the programme it is assumed that $\tilde{g}/g$ differs from unity only very near the surface. Since in this region $q = m/M$ is one with very high precision, $\tilde{q} = x^3 \tilde{A}_1$ can be taken as a very good approximation to $\tilde{g}/g$, allowing, *e.g.*, the evaluation of V_g and A.

The dimensionless perturbation variables $y_1 - y_4$ are defined as in Section 2.1; in particular the actual gravitational acceleration g is used in the definition of y_3 (*cf.* equation 2.5a).

With these definitions, the equations satisfied by the y_i are, for non-radial oscillations

$$
x\frac{\mathrm{d}y_1}{\mathrm{d}x} = (\tilde{V}_g - 2)y_1 + \left(1 - \frac{\tilde{V}_g}{\tilde{\eta}}\right) y_2 - V_g y_3 \;, \tag{A.18}
$$

$$
x\frac{\mathrm{d}y_2}{\mathrm{d}x} = [l(l+1) - \tilde{\eta}\tilde{A}]y_1 + (\tilde{A} - 1)y_2 + \eta\tilde{A}y_3 \;, \tag{A.19}
$$

$$
x\frac{\mathrm{d}y_3}{\mathrm{d}x} = y_3 + y_4 \;, \tag{A.20}
$$

and

$$
\begin{aligned}
x\frac{\mathrm{d}y_4}{\mathrm{d}x} = &- \lambda\tilde{A}Uy_1 - \lambda U\frac{\tilde{V}_g}{\tilde{\eta}}y_2 \\
&+ [l(l+1) + U(A-2) + (1-\lambda)UV_g]y_3 + 2(1-U)y_4 \;.
\end{aligned}
\tag{A.21}
$$

Here $\tilde{\eta} = l(l+1)\tilde{g}/(\omega^2 r)$.

For radial oscillations the equations are

$$
x\frac{\mathrm{d}y_1}{\mathrm{d}x} = (\tilde{V}_g - 2)y_1 - \tilde{V}_g\frac{\sigma^2 x^2}{\tilde{q}}y_2 \;, \tag{A.22}
$$

and

$$
x\frac{\mathrm{d}y_2}{\mathrm{d}x} = \left[x - \frac{1}{\sigma^2 x^2}(\tilde{q}\tilde{A} - \lambda q U)\right] y_1 + \tilde{A}y_2 \;. \tag{A.23}
$$

To use this option, the equilibrium model must obviously be set up correctly, as defined in equations (A.16). In addition, the parameter `iturpr` must be set to 1.

A.3.2 Neglect of the Eulerian perturbation in the turbulent pressure (`iturpr = 2`*)*

Here we assume that the pressure can be written as

$$p = p_{\rm g} + p_{\rm turb} \; , \tag{A.24}$$

where $p_{\rm g}$ is the thermodynamic pressure and $p_{\rm turb}$ is the turbulent pressure. Hence, the equations of motion are written as

$$\rho \frac{\rm D\mathbf{u}}{{\rm D}t} = -\nabla p + \rho \mathbf{g} \; , \tag{A.25}$$

with no explicit turbulent body force. In the equilibrium model, p is assumed to be in hydrostatic equilibrium, so that the equation of hydrostatic support is

$$\frac{{\rm d}p}{{\rm d}r} = -g\rho \; . \tag{A.26}$$

In the oscillation calculation the assumption is now that the Eulerian perturbation of $p_{\rm turb}$ can be neglected. Hence the Eulerian perturbation of equation (A.25) may be written

$$\rho \frac{\partial^2 \delta \mathbf{r}}{\partial t^2} = -\nabla p_{\rm g}' - \rho' g \mathbf{a}_r + \rho \nabla \Phi' \; . \tag{A.27}$$

The condition of adiabaticity is assumed to relate the Lagrangian perturbation of the gas pressure to the density perturbation,

$$\frac{\delta p_{\rm g}}{p_{\rm g}} = \Gamma_1 \frac{\delta \rho}{\rho} \; , \tag{A.28}$$

where again Γ_1 is the thermodynamic adiabatic exponent.

In the oscillation equations, derived under these assumptions, we need to distinguish between the gradients in $p_{\rm g}$ and p. Thus we introduce

$$V_g^{(0)} = -\frac{1}{\Gamma_1} \frac{{\rm d}\ln p_{\rm g}}{{\rm d}\ln r} \; , \qquad V_g^{(1)} = -\frac{1}{\Gamma_1} \frac{{\rm d}\ln p}{{\rm d}\ln r} = \frac{\rho g r}{\Gamma_1 p} \; , \tag{A.29}$$

and

$$A^{(0)} = -V_g^{(0)} - \frac{{\rm d}\ln \rho}{{\rm d}\ln r} \; , \qquad A^{(1)} = -V_g^{(1)} - \frac{{\rm d}\ln \rho}{{\rm d}\ln r} \; . \tag{A.30}$$

This requires an extension of the model format defined in Section 5. Specifically, we replace the definitions of A_2 and A_4, and add a new variable A_6, in equation (5.1):

$$\begin{aligned} A_2 &= V_g^{(1)} \; , \\ A_4 &= A^{(1)} \; , \\ A_6 &= V_g^{(0)} - V_g^{(1)} = -(A^{(0)} - A^{(1)}) \; , \end{aligned} \tag{A.31}$$

the remaining definitions being unchanged. To flag for a model file of this format, the otherwise unused variable `data(8)` should be set to 10.

The dimensionless perturbation variables $y_1 - y_4$ are defined as in Section 2.1.

With these definitions, the equations satisfied by the y_i are, for non-radial oscillations (note that for the time being I have not considered the option of $\lambda \neq 1$ in this case)

$$x \frac{{\rm d}y_1}{{\rm d}x} = (V_g^{(0)} - 2) y_1 + \left(1 - \frac{V_g^{(1)}}{\eta} \right) y_2 - V_g^{(1)} y_3 \; , \tag{A.32}$$

$$x\frac{\mathrm{d}y_2}{\mathrm{d}x} = [l(l+1) - \eta A^{(0)}]y_1 + (A^{(1)} - 1)y_2 + \eta A^{(1)} y_3 \ , \tag{A.33}$$

$$x\frac{\mathrm{d}y_3}{\mathrm{d}x} = y_3 + y_4 \ , \tag{A.34}$$

and

$$\begin{aligned} x\frac{\mathrm{d}y_4}{\mathrm{d}x} = & - A^{(0)}U y_1 - U\frac{V_g^{(1)}}{\eta}y_2 \\ & + [l(l+1) + U(A^{(1)} - 2)]y_3 + 2(1-U)y_4 \ . \end{aligned} \tag{A.35}$$

For radial oscillations the equations are

$$x\frac{\mathrm{d}y_1}{\mathrm{d}x} = (V_g^{(0)} - 2)y_1 - V_g^{(1)}\frac{\sigma^2 x^2}{q}y_2 \ , \tag{A.36}$$

and

$$x\frac{\mathrm{d}y_2}{\mathrm{d}x} = \left[x - \frac{q}{\sigma^2 x^2}(A^{(0)} - U)\right] y_1 + A^{(1)} y_2 \ . \tag{A.37}$$

To use this option, the equilibrium model must obviously be set up correctly, as defined in equations (A.31). In addition, the parameter `iturpr` must be set to 2.

A.3.3 Neglect of the Lagrangian perturbation in the turbulent pressure

We still assume the separation in pressure given in equation (A.24), the equations of motion (A.25) and that the equilibrium total pressure satisfies the equation of hydrostatic support (A.26). However, we now assume that the Lagrangian perturbation $\delta p_{\rm turb} = 0$ (see also Rosenthal *et al.* 1995). The equations of motion are now

$$\rho\frac{\partial^2 \delta \mathbf{r}}{\partial t^2} = -\nabla p' - \rho' g \mathbf{a}_r + \rho \nabla \Phi' \ . \tag{A.38}$$

Furthermore, assuming still equation (A.28) for the adiabatic change in the *thermodynamic* pressure, we obtain for the relation between the Lagrangian perturbations in the total pressure and density:

$$\frac{\delta p}{p} = \frac{\delta p_{\rm g}}{p} = \frac{\delta p_{\rm g}}{p} = \frac{p_{\rm g}}{p}\Gamma_1 \frac{\delta \rho}{\rho} \ , \tag{A.39}$$

or

$$\frac{\delta p}{p} = \Gamma_1^{\rm r} \frac{\delta \rho}{\rho} \ , \tag{A.40}$$

defining *the reduced* $\Gamma_1^{\rm r} \equiv (p_{\rm g}/p)\Gamma_1$ (Rosenthal *et al.* 1995). Here, obviously, Γ_1 is the thermodynamic adiabatic exponent.

With this definition, the oscillation equations are precisely as in the case without turbulent pressure, when expressed in terms of the Eulerian perturbation p' in total pressure, provided Γ_1 is replaced throughout by $\Gamma_1^{\rm r}$. It should be noticed also that this is the case for *any* relation of the form (A.40), for a function $\Gamma_1^{\rm r}$ defined in the equilibrium model. Indeed, it was argued by Rosenthal *et al.* (1997) that a more appropriate form of $\Gamma_1^{\rm r}$ could be obtained by averaging the equations for turbulent convection.

Implementation of this approximation therefore only requires modification of the equilibrium model. Specifically, the definitions of $A_2 - A_4$ are changed to

$$\begin{aligned} A_2 &= -\frac{1}{\Gamma_1^{\mathrm{r}}} \frac{\mathrm{d} \ln p}{\mathrm{d} \ln r} = \frac{G m \rho}{\Gamma_1^{\mathrm{r}} p r} \, , \\ A_3 &= \Gamma_1^{\mathrm{r}} \, , \\ A_4 &= \frac{1}{\Gamma_1^{\mathrm{r}}} \frac{\mathrm{d} \ln p}{\mathrm{d} \ln r} - \frac{\mathrm{d} \ln \rho}{\mathrm{d} \ln r} \, , \end{aligned} \qquad (A.41)$$

with the remaining definitions being unchanged.

The definition of the sound speed in this case deserves a little comment. It is evident from the unchanged form of the oscillation equations that the relevant expression for the sound speed is

$$c^2 = \frac{\Gamma_1^{\mathrm{r}} p}{\rho} \, , \qquad (A.42)$$

expressed in terms of the modified Γ_1 and the total pressure. It is interesting that with the simple definition of Γ_1^{r} implied by equation (A.39) this is identical to $\Gamma_1 p_{\mathrm{g}}/\rho$, *i.e.*, the “thermodynamic” squared sound speed. However, with a more general definition of Γ_1^{r} this is not generally the case. On the other hand, it is obvious that equation (5.7) still holds.