Notes on using the solar models and revised adiabatic pulsations package

July 11, 2012

J. Christensen-Dalsgaard

1 Introduction

These notes give some information on the programs and data provided with the package. It is assumed that the package has been installed as described in the `README` file provided, and that the reader is familiar with the basic concepts discussed in *Notes on adiabatic oscillation programme* (in the following *Notes*), particularly the input format and output files. All directory names are given relative to the root where the package was unpacked and installed.

The basic way that all the programs are run is through `csh` scripts provided in the directory `bin`; to get easy access to the scripts, `bin` (with its appropriate prefix) must be included in the command search path. These call the executables (denoted by ending in `.x`). Parameters to the executable may be provided in one of two ways:

- as arguments to the script
- in a *control file* which is provided as as argument to the script; this is typically used in more complex cases. The control file sets up parameters and input and output files for the program.

In most cases the proper arguments to the scripts, with a brief description, is provided by calling the script either with no argument or with the argument `-help`,

```
script -help
```

Sample control files are provided in the tar files setting up the test cases (see the `README` file). The structure for these control files is similar to the structure of the control file to the basic pulsation code, as described in *Notes*, section 7.2.1.

A note on program naming: for ease of reference I have in almost all cases kept the same names as in my own installation. In particular, I have kept the "`.d`" in the most of the names, indicating that the programs are in double precision (8 byte reals).

1.1 Notes on revisions

The present set of programs replaces the set released originally in 1994. There have been numerous changes in several of the programs and corresponding control files; for this reason, unfortunately, backwards compatibility has not been ensured. However, none of the functionality in the old package has been lost. Also, the formats of the basic data products are the same, so that there should be no difficulty in using the new package with old files of modes or models. It is suggested that the user completely replaces the old package with the new, rather than to try to combine them.

The present notes are provided for the beta release V0_1 of the package and are not yet complete. In particular, more details about the test cases are so far provided in the `README` file, which should be given preference, in case of conflict.

2 Computing adiabatic modes

The program is started by the command

```
adipls.c.d <control file>
```

Test cases, including the required control files, are provided for a solar model, a model of a somewhat more massive stars, and two red-giant models. In the case of the solar model, the sample control file recomputes a modest set of modes, at selected degrees, based on reading trial frequencies from a grand summary. Only eigenfrequencies are output. For the other cases frequencies are found by scanning in frequencies. These cases are useful for testing that the program works correctly. The computed frequencies can be compared with the original frequencies, also provided, by using the program `freqdif.d` (see below). It will presumably be comparatively easy to modify the control files to consider other cases. A full list of control parameters is provided in *Notes*.

If pulsations are to computed for other models than those provided, the model must be transformed into the appropriate format, as described in *Notes*, Sections 5 and 7.1. Note in particular that for a physical model (in contrast to a polytrope) D_7 must be set to -1 (this affects the way the surface boundary condition is treated).

In general, the mesh resulting from, *e.g.*, an evolution calculation is not optimal for oscillation calculations. The mesh can be reset with the program `redistrb.c.d` (or the previous code, `redistrb.d`, now largely obsolete, which is still used for the solar model for consistency with the previous calculations), described below.

3 Auxiliary programs

The package contains a set of programs for manipulating the output of the pulsation program and the model files. These are provided in the directory `adiajobs` as Fortran source code, and are compiled and linked with the `make` command (see the `README` file). The following nomenclature is used, for any program `program`:

- `program.d.f` is the Fortran source code.
- `program.d.x` is the executable program.
- `program.d` is the `csh` script (residing in `bin`) used to call the program.

Unlike the pulsation code, there are no separate notes documenting the use of these programs. Below follows a brief list of the most important of them, with indications of what they do and how they are called. In general, a fairly detailed description of the input parameters and operation of the program is provided in the source code.

The most commonly used sample control files are provided in the directories of the test cases. Other examples of control files can be found in the directory `.../adiajobs/in`.

- `form-amdl.d`: Transforms between binary and ASCII versions of models for pulsation program. It is called with the command

  ```
  form-amdl.d <case> <input file> <output file>
  ```

  Here `<case>` determines the direction of the transformation:

  - `case = 1`: transform from binary to ASCII
  - `case = 2`: transform from ASCII to binary

- `form-agsm.d`: Transforms between binary and ASCII versions of grand summaries. It is called with the command

  ```
  form-agsm.d <case> <input file> <output file> [diag]
  ```

  Here `<case>` determines the direction of the transformation:

  - `case = 1`: transform from binary to ASCII
  - `case = 2`: transform from ASCII to binary

  `[diag]` controls the amount of output, as in the `scan-agsm.d` command (see below).

- `redistrb.c.d`: Resets mesh in model for pulsation program. It is called with the command

  ```
  redistrb.c.d <control file>
  ```

  Sample control files (`redistrb.in` and `redistrb.c.in`) are provided. In the case of the solar models, these are set up to set meshes suitable for p- and g-mode calculations. Although it is possible to set individual parameters, the following two choices should be appropriate for computing modes in solar models:

  - `icase = 11` for p modes.
  - `icase = 12` for g modes.

  In general, I use the convention of denoting the redistributed models (and other output files) with 600, 1200, 2400, 4800, ... points by the file trailers `<mcase>, <mcase>1, <mcase>2, <mcase>3`, ... For the original p- and g-mode meshes `<mcase>` was `p` and `g`, respectively. In the ongoing update of the mesh scheme I use `pxo` and `prxo`, respectively, for `<mcase>`, the latter indicating a computing scheme appropriate for red-giant models.

- `scan-amdl.d`: prints summary of pulsation model file. It is called with the command

  ```
  scan-amdl.d <pulsation model file>
  ```

- `set-asscal.d`: Adds dimensionless integral of $\mathrm{d}r/c$ and dimensionless asymptotic scaling factor to pulsation model file. See Appendix A for precise definitions. It is called with the command

  ```
  set-asscal.d <input model file> <output model file> [truncation radius]
  ```

  where

  - `truncation radius` (default photospheric radius) is upper limit of scaling integral.

- `compamod.n.d`: Finds differences between two pulsation-model files. It is called with the command

  ```
  compamod.n.d <control file>
  ```

  A sample control file (`compamod.n.in`) is provided.

- `fgong-amdl.d`: Converts model in ASCII GONG format (see Section 5) to pulsation model file. It is called with the command

```
fgong-amdl.d <Input GONG file> <Output pulsation-model file>
```

- `diff-fgong.d`: Finds differences between two models in ASCII GONG format (see Section 5). It is called with the command

  ```
  diff-fgong.d <First input model> <Second input model> <Output file> [case]
  ```

  Use `diff-fgong.d -help` to get details about the different cases. The default is difference at fixed fractional radius r/R.

- `scan-agsm.d`: prints summary of grand summary file. It is called with the command

  ```
  scan-agsm.d <grand summary file>
  ```

  or

  ```
  scan-agsm.d <grand summary file> 1
  ```

  In the first case all modes are listed, in the second only the first mode for each degree.

- `scan-ssm.d`: prints summary of short summary file. It is called with the command

  ```
  scan-ssm.d <short summary file>
  ```

- `scan-amde.d`: prints summary of eigenfunction file. It is called with the command

  ```
  scan-amde.d <eigenfunction file> <case>
  ```

  Here `case = 1` for a file containing the full set of eigenfunctions, `case = 2` for file containing restricted set (for formats for eigenfunction files, see *Notes*, section 8.5).

- `set-obs.d`: sets ASCII formatted file of mode data from grand or short summary, with each record having the form l, n, ν, where ν is frequency in μHz. (This can be thought of as frequencies in the form of observed data.) It is called with the command

  ```
  set-obs.d <case> <input file> <output file>
  ```

  Here the input file can be either a grand summary or a short summary. `<case>` depends on the type of input file and, for input from a grand summary, determines the choice of frequency:

  - `case = 1`: grand summary, variational frequency.
  - `case = 2`: short summary.
  - `case = 4`: grand summary, from eigenfrequency in `cs(20)`. Note that this allows setting Cowling approximation frequency.
  - `case = 5`: grand summary, from Richardson extrapolation frequency in `cs(37)`, if this is set.
  - `case = 6`: grand summary, from (possibly corrected) eigenfrequency in `cs(21)`.
  - Otherwise variational frequency is used.

  If `case > 10`, set frequencies according to `case - 10`, but include also mode energy (*cf. Notes*, section 4.3) in the file.

- `diff-sum.d`: Compares two mode sets, assumed to be ordered properly, outputting modes that appear in one set but not the other. The mode sets can be of the form of grand or short summaries, or observed data. It is called with the command

```
diff-sum.d <case 1> <case 2> <input file 1> <input file 2> \
                <excess 1> <excess 2>
```

Here `case 1` and `case 2` determine the types of mode sets (which may be different):

- `case = 1`: grand summary.
- `case = 2`: short summary.
- `case = 3`: observed frequencies.

Also

- `<excess 1>`: Output file of modes in file 1 but not in file 2
- `<excess 2>`: Output file of modes in file 2 but not in file 1

Note: `excess` files are output in same form as corresponding input file.

- `res-amde.d`: sets restricted eigenfunction file from full file. It is called with the command

  ```
  res-amde.d <input file> <output file> <case>
  ```

  Here `case = 1` produces an output file with y_1, y_2, whereas `case = 2` produces an output file with $\hat{z}_1, \hat{z}_2$ (for formats for eigenfunction files, see *Notes*, section 8.5).

- `freqdif.d`: Computes frequency differences between various types of frequency sets. It is called with the command

  ```
  freqdif.d <control file>
  ```

  Sample control files (`freqdif.in`) are provided, to compute differences between the original modes provided in a grand summary, and the modes computed with the control files for the adiabatic pulsation program.

  The frequency sets may be in the form of grand or short summaries, or observed frequencies, as specified by the appropriate input parameters (a rudimentary set of notes is provided in `freqdif.in`). `freqdif.d` automatically selects the overlapping modes between the two sets. If the first mode set is in the form of a grand summary, the frequency differences can be automatically scaled by the mode inertia ratio Q_{nl}.

- `selsum.d`: Selects a subset of a mode set, determined by windowing and/or by matching modes in another set. Both input and selection sets can, independently, be of the form of grand or short summaries, or observed data. It is called with the command

  ```
  selsum.d <control file>
  ```

  A sample control file (`selsum.in`) is provided.

- `mer-sum.d`: Combines two mode sets, assumed to be ordered properly. The mode sets can be of the form of grand or short summaries, or observed data. It is called with the command

  ```
  mer-sum.d <case> <input file 1> <input file 2> <output file>
  ```

  Here `case` determines the type of mode set:

  - `case = 1`: grand summary.
  - `case = 2`: short summary.
  - `case = 3`: observed frequencies, without errors.

  - `case = 4`: observed frequencies, with errors.

- `sortsum.d`: Combines and reorders several mode sets, possibly applying windowing to them. The mode sets can be of the form of grand or short summaries, or observed data. It is called with the command

  ```
  sortsum.d <control file>
  ```

  A sample control file (`sortsum.in`) is provided. This is typically useful if several runs of the pulsation code is required to produce a complete mode set. To merge two mode sets, already ordered, the simpler program `mer-sum.d` can be used (see above).

- `setexec.d`: Scans for missing modes in a mode set file, estimating their frequencies from interpolation, and writing the estimated mode parameters as a short summary file on unit 3. This may subsequently be used as trial input for a new attempt to determine the modes. The input file may be in the form of a short summary or grand summary. The pogramme is called with the command

  ```
  setexec.d <control file>
  ```

  A sample control file (`setexec.in`) is provided.

It is obvious that this is only a sample of the programmes supplied in `adiajobs`. The user is invited to explore.

4 IDL procedures

The package contains a small set of IDL procedures to read products of the pulsation code. They are in the directory `idl_pro`. Some documentation on their use is contained at the start of each procedure.

The following procedures are provided:

- `read_amde.pro`: reads an eigenfunction from eigenfunction file.
- `read_amdes.pro`: reads several eigenfunctions from eigenfunction file.
- `read_amdl.pro`: reads pulsation model from file.
- `read_gsm.pro`: reads a single grand summary from file.
- `read_gsms.pro`: reads all grand summaries on file.
- `read_fgong.pro`: reads model in ASCII GONG format (see below).

5 Models

The `models.c.tar`, `star.c.tar` and `red-giant.c.tar` distribution files contain calibrated models of the present Sun, a model of a $1.5\,\mathrm{M}_\odot$ star, and models of two red giants, respectively, frequencies for these models and input control files for the programmes. (See the `README` file for details.)

5.1 Some notes on naming conventions

I have kept my internal notation for the files, even though this may, in isolation, seem rather artificial. This involves a fixed naming convention for products of the computation. This is obviously arbitrary, but provides an immediate overview of the results:

- `amdl.<model descriptor>`: Pulsation-model file on the original mesh used in the evolution calculation. The variables and file format are described in the Appendix.
- `amdl.<model descriptor>.<mcase><n>`: Pulsation-model file on a redistributed mesh; `<mcase>1`, `<mcase>2`, `<mcase>3`, ... indicate meshes with 1200, 2400, 4800, ... points. Simple meshes suitable for p- and g-mode calculations are indicated by `<mcase> = p` and `g`, respectively.
- `fgong.<model descriptor>`: extensive set of model variables, in the standard ASCII GONG format. The variables, and the format, are described in the notes contained in the file `file-format.tex` in the `notes` directory.

- `agsm.<model descriptor>.<mcase><n>`: grand summary of modes computed for model `amdl.<model descriptor>.<mcase><n>`. Different sets are distinguished by adding yet another trailer.
- `ssm.<model descriptor>.<mcase><n>`: short summary of modes computed for model `amdl.<model descriptor>.<mcase><n>`.
- `amde.<model descriptor>.<mcase><n>`: full eigenfunction file computed for model `amdl.<model descriptor>.<mcase><n>`.
- `amde.<model descriptor>.<mcase><n>.y`: restricted eigenfunction file, providing y_1, y_2, computed for model `amdl.<model descriptor>.<mcase><n>`.
- `amde.<model descriptor>.<mcase><n>.z`: restricted eigenfunction file, providing $\hat{z}_1, \hat{z}_2$, computed for model `amdl.<model descriptor>.<mcase><n>`.

ASCII versions of binary files (see the `form-amdl.d` and `form-agsm.d` commands above) are indicated by adding “`.for`” to the names of the binary files.

5.2 Notes on the models and frequencies

In the `models.c.tar` distribution, the results are for Model S, used as reference in the GONG series of *Science* articles (see Christensen-Dalsgaard *et al.* 1996). Full descriptions of the physics used in the calculations is given in the paper. For the other test cases, see the `README` file.

The models are provided in terms of pulsation variables (`amdl` models) and ASCII GONG variables (`fgong` models). They are identified by my internal model descriptors. This naming may appear a little unwieldy; however, I strongly recommend recording these names for later unique reference to the models. The following case is provided `models.tar`:

- `l5bi.d.15`: Model S of Christensen-Dalsgaard *et al.* (1996). OPAL(1992) opacity, OPAL equation of state, including helium and heavy-element settling. Age of present Sun 4.6 Gyr.

In addition to the model file on the original mesh, I have also included the model on a p-mode mesh with 2400 points (`amdl.l5bi.d.15.p2`).

Files of oscillation results:

- `agsm.l5bi.d.15.p2`: Grand summary of a moderate set of modes, at selected degrees (use `scan-agsm.d` to find out which). This set is useful for testing the code and initial investigations of the effects of changes in the model.
- `agsm.l5bi.d.15.p2.md`: Grand summary of an extensive set of modes, including all p modes below the acoustic cut-off frequency at degrees below 300. This is probably adequate for analysis of the GONG data (but, as *Notes* will show, there are easy ways to produce complete sets to even higher degree).
- `obs.l5bi.d.15.p2.md`: ASCII file ("observational format") for an extensive set of modes, including all p modes below the acoustic cut-off frequency at degrees below 300.
- `agsm.l5bi.d.15.g3`: Grand summary of a fairly extensive set of g and f modes of degree 1 – 7.

References

Christensen-Dalsgaard, J., Däppen, W., Ajukov, S. V., Anderson, E. R., Antia, H. M., Basu, S., Baturin, V. A., Berthomieu, G., Chaboyer, B., Chitre, S. M., Cox, A. N., Demarque, P., Donatowicz, J., Dziembowski, W. A., Gabriel, M., Gough, D. O., Guenther, D. B., Guzik, J. A., Harvey, J. W., Hill, F., Houdek, G., Iglesias, C. A., Kosovichev, A. G., Leibacher, J. W., Morel, P., Proffitt, C. R., Provost, J., Reiter, J., Rhodes Jr., E. J., Rogers, F. J., Roxburgh, I. W., Thompson, M. J., Ulrich, R. K., 1996. [The current state of solar modeling]. *Science*, **272**, 1286 – 1292.

Appendix. The structure of the pulsation model file

For convenience, I repeat the information about the model-file structure here (see also *Notes*, Section 5). In addition, I here list the additional variables set up by calling `set-asscal.d`:

The model variables is defined in the array `data(1:8)` of global parameters, `x(1:nn)` defining the mesh and `aa(1:ia, 1:nn)` giving the variables at each mesh point. The basic set of variables consists of

$$\begin{aligned}
\texttt{x(n)} &= x \equiv r/R \;, && (A.1)\\
\texttt{aa(1,n)} &= A_1 \equiv q/x^3, \qquad \text{where } q = m/M \;, && (A.2)\\
\texttt{aa(2,n)} &= A_2 = V_g \equiv -\frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} = \frac{Gm\rho}{\Gamma_1 p r} \;, && (A.3)\\
\texttt{aa(3,n)} &= A_3 \equiv \Gamma_1 \;, && (A.4)\\
\texttt{aa(4,n)} &= A_4 = A \equiv \frac{1}{\Gamma_1}\frac{\mathrm{d}\ln p}{\mathrm{d}\ln r} - \frac{\mathrm{d}\ln \rho}{\mathrm{d}\ln r} \;, && (A.5)\\
\texttt{aa(5,n)} &= A_5 = U \equiv \frac{4\pi\rho r^3}{m} \;. && (A.6)
\end{aligned}$$

In addition, in some models

$\texttt{aa(6,n)}$ is a variable dealing with turbulent pressure .

In models extended with `set-asscal.d`

$$\begin{aligned}
\texttt{aa(7,n)} &= A_7 \equiv \int_0^x \frac{\mathrm{d}x'}{\tilde c(x')} \;, && (A.7)\\
\texttt{aa(8,n)} &= A_8 \equiv \int_x^{x_s} \left[1 - \left(\frac{\tilde c(x')x}{\tilde c(x)x'}\right)^2\right]^{-1/2} \frac{\mathrm{d}x'}{\tilde c(x')} \;. && (A.8)
\end{aligned}$$

Here $\tilde c = (R/GM)^{1/2} c$ is the dimensionless sound speed. The quantities $A_1 - A_8$ are clearly all dimensionless. In addition we use the following "global" quantities for the model:

$$\begin{aligned}
\texttt{data(1)} &= D_1 \equiv M \;,\\
\texttt{data(2)} &= D_2 \equiv R \;,\\
\texttt{data(3)} &= D_3 \equiv p_{\rm c} \;,\\
\texttt{data(4)} &= D_4 \equiv \rho_{\rm c} \;,\\
\texttt{data(5)} &= D_5 \equiv -\left(\frac{1}{\Gamma_1 p}\frac{\mathrm{d}^2 p}{\mathrm{d}x^2}\right)_{\rm c} \;, \qquad (A.9)\\
\texttt{data(6)} &= D_6 \equiv -\left(\frac{1}{\rho}\frac{\mathrm{d}^2 \rho}{\mathrm{d}x^2}\right)_{\rm c} \;,\\
\texttt{data(7)} &= D_7 \equiv \mu \;,\\
\texttt{data(8)} &= D_8 : \text{see below} \;.
\end{aligned}$$

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 $p_{\rm c}$ and $\rho_{\rm c}$ 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 physical model D_7 is set to -1; in a model with a polytropic surface layer, D_7 is the polytropic index of the surface region. 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.

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 options are

- $D_8 = 10$: the file contains 6 variables $A_1 - A_6$,
- $D_8 = 100$: the file contains 8 variables $A_1 - A_8$,

as defined above. The value of D_8 is checked when the file is opened for read; hence the structure must be the same for all models in a given file.