

InterpolateModel

Version 1.0

Daniel Reese

November 2015

Contents

1	Getting started	3
1.1	Running the program	3
1.2	Using the program	3
2	Model interpolation	8
3	Linear interpolation between two models	8
3.1	Acoustic variables	8
3.2	The radial grid	9
4	File formats	10
4.1	Input file for batch mode	10
4.2	Output files from batch mode	10
4.2.1	Interpolated models	11
4.2.2	A list of interpolated models	11
4.2.3	Summary result files	11
4.3	Stellar models	12
4.3.1	MOD (binary/ascii)	13
4.3.2	CLES format	14
4.3.3	LOSC format	14
4.3.4	Comparison of different model formats	15
4.3.5	Useful formulas	15
4.4	GZIP compression	16
5	Known bugs	16
6	Copyright notices	16
6.1	The Dierckx.java library	16
6.2	Source code for reading fortran binary files	16
6.3	Supplementary notices	17

Acknowledgements

This program was written by Daniel Reese, whose work was supported by the SPACEINN network, a major international collaboration funded by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 312844.

1 Getting started

1.1 Running the program

`InterpolateModel` runs under Java 7.0 or later versions. If Java is not installed on your computer, or is not sufficiently up-to-date, it can be downloaded from:

<http://www.java.com/en/>

JRE (Java Runtime Environment) allows you to run Java programs but not to compile your own. JDK (Java Development Kit) allows you to run and compile Java programs.

To run the program download and extract the file `InterpolateModel.jar` from the following website:

<http://bison.ph.bham.ac.uk/spaceinn/interpolatemodel/index.html>

then type the following command in a command window, in the directory that contains `InterpolateModel.jar`:

```
java -jar InterpolateModel.jar
```

If you want to to run `InterpolateModel` in batch mode simply type:

```
java -jar InterpolateModel.jar input_file
```

where `input_file` is the name of your input file (its format will be described in a later section).

If for some reason you need to do calculations which require a large amount of memory, for example 4000 MB, then use the following command:

```
java -Xmx4000m -jar InterpolateModel.jar
```

Note: the option `-Xmx` is nonstandard and may change according to the release installed on your computer.

1.2 Using the program

`InterpolateModel`, which has been derived from `InversionKit`, can load multiple models simultaneously and interpolate between these models, in a way that's compatible with the AIMS code. There are two main ways of doing this:

1. **in batch mode:** this is the preferred way of doing things as it can produce multiple interpolated models, and the output is compatible with `InversionPipeline`. This is done by running `InterpolateModel` with an input file as an argument. The format of this input file will be described in Sect. 4.1. The AIMS code has been set up to produce input files with this format.
2. **interactively:** this can be useful for examining specific cases manually, and overplotting structural files from multiple models. However, it can only produce one in-

terpolated model at a time, and some of the information needed for `InversionPipeline` is lost along the way (as it is supplied by the input file from the previous approach).

When run in interactive mode, the program contains a window with the following two tabs:

- **Stellar model:** this allows the user to load and visualise multiple models, do basic manipulations on these models, as well as interpolate between these models via linear combinations.
- **Physical constants:** this allows the user to modify physical constants (such as the gravitational constant) as well as astronomical constants (such as the solar mass)

The next few pages show a set of screen captures along with a brief description of the various buttons and options which appear in the different tabs. The actual appearance of these tabs and windows may vary from one platform to another depending on the Java installation. This section can therefore be viewed as a quick and easy guide for `InterpolateModel`. For more detailed information on the relevant file formats as well as some technical aspects, we refer the reader to the following sections.

InterpolateModel 1.0

Stellar model Physical constants

U η mass gravity $d(\text{gravity})/dr$ $(1-\rho/\rho_0)/x^2$ V_g

ρ $d(\rho)/dr$ P Γ_1 c^2 $(dc/dr)/r$ $c(\tau)$ $c'(\tau)$ N^2 A

IDL TXT AS DUPL ☐ x-log ☐ y-log

Density

Density (in g/cm^3)

Radius, r/R

Mass: 1.499 M_\odot Radius: 1.516 R_\odot
 Ω_K : 6.553e+01 μHz G : 6.67428e-08 (cgs)
 Number of points: 1638

☒ model.mdl ☒ model.cles ☒ model.mod

Load model
 Load model from internet
 Load HELAS model
 Generate Polytrope
 Save model
 Clear model
 Overplot models
 Find G from model
 Rescale model
 Remesh model
 Remove discontinuities
 Interpolate between models
 Native Endian = Little
☒ show double points
☐ show grid points

load a stellar model from a file
 load a stellar model an URL
 load a model from the CoRoT/ESTA HELAS website
 generate a polytropic model
 save the current model
 remove the current model
 plot the structural profiles from all the loaded models together
 recalculate gravitational constant based on hydrostatic equilibrium
 rescale the mass and radius of the current model
 interpolate model to new grid
 remove discontinuities from the model
 interpolate between multiple models (as described later on)
 tabbed pane with various structural profiles
 right click on plots to open dialog on selected grid point
 information on the current model
 tabbed pane with various models

InterpolateModel 1.0

Stellar modelPhysical constants

Gravitational constant, G

Current value: 6.674280000000000e-08 (cm³·g⁻¹·s⁻²)

	Update value
G = 6.6716823×10 ⁻⁸ (GraCo, OSCROX, POSC, PULSE, CoRoT/ESTA models)	
G = 6.67232×10 ⁻⁸ (ADIPLS, FILOU, LOSC, LNAWENR, Model S)	
G = 6.67259×10 ⁻⁸ (NOSC)	
G = 6.67384×10 ⁻⁸ (2010 CODATA)	
G = 6.67428×10 ⁻⁸ (MESA)	

Solar mass, M_☉

Current value: 1.989190000000000e+33 (g)

	Update value
M _☉ = 1.98919×10 ³³ (CoRoT/ESTA models)	
M _☉ = 1.989×10 ³³ (Model S)	
M _☉ = 1.98855×10 ³³ (=GM _☉ /G, where GM _☉ comes from Cox 2000, and G from 2010 CODATA)	

Solar radius, R_☉

Current value: 6.959900000000000e+10 (cm)

	Update value
R _☉ = 6.9599×10 ¹⁰ (CoRoT/ESTA models, Model S)	
R _☉ = 6.95613×10 ¹⁰ (Haberreiter et al. 2008)	

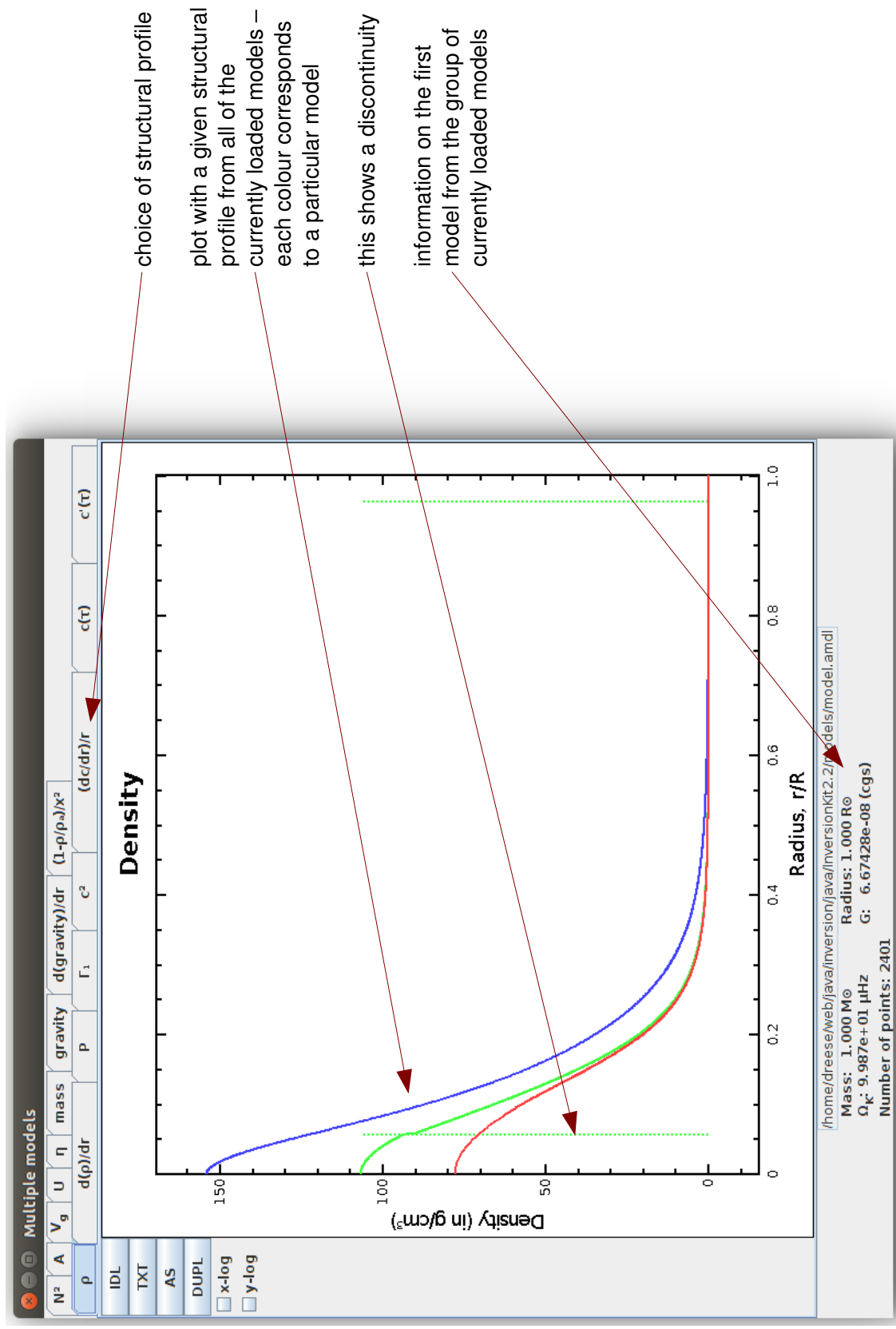
Save constants

this text field, and these buttons can be used to adjust the value of the gravitational constant in InterpolateModel (which intervenes in the interpolated models)

this text field, and these buttons can be used to adjust the value of the solar mass, in InterpolateModel

this text field, and these buttons can be used to adjust the value of the solar radius, in InterpolateModel

this saves the values of the above constants for future sessions with InterpolateModel



2 Model interpolation

The following section briefly describes model interpolation as carried out in `InterpolateModel` and explains how it relates to results from the `AIMS` code.

When considering interpolation between models, there are two very important aspects to consider:

1. how does one actually combine two or more models?
2. how does one obtain the appropriate interpolation coefficients?

`InterpolateModel` only addresses the first question by combining the acoustic structure of models. The `AIMS` code deals with the second question. `AIMS` also partially deals with the first question in the sense that it combines the global parameters and frequencies of models, but it does not combine their acoustic structure. However, care has been taken so that when `AIMS` combines the global parameters, it does so in a way that's consistent with the way `InterpolateModel` combines the acoustic structure. Hence, the two codes should be used in conjunction.

3 Linear interpolation between two models

Let us start by considering two models, Models 1 and 2. We wish to interpolate between the two, thereby producing a third model, Model 3. Specifically, Model 3 will be a weighted “sum” of Models 1 and 2, the coefficients being a and b , respectively, where $a + b = 1$. In what follows, we will use the indices 1, 2 and 3 to designate these models.

3.1 Acoustic variables

Let R_i be the photospheric radii of the different models. `InterpolateModel` combines the models by considering that the density and Γ_1 profiles of Model 3 is simply the weighted sum of these profiles from Models 1 and 2. This leads to:

$$\rho_3(xR_3) = a\rho_1(xR_1) + b\rho_2(xR_2), \quad (1)$$

$$\Gamma_{1,3}(xR_3) = a\Gamma_{1,1}(xR_1) + b\Gamma_{1,2}(xR_2), \quad (2)$$

where $x \in [0, 1]$. Furthermore, we want to make sure that if the masses of Models 1 and 2 are identical, then the mass of Model 3 is also the same. A simple way of achieving this is by imposing:

$$M_3 = aM_1 + bM_2 \quad (3)$$

If we combine Eqs. (1) and (3) this uniquely defines the radius of Model 3:

$$\begin{aligned} M_3 &= R_3^3 \int_0^1 4\pi\rho_3(xR_3)x^2 dx \\ &= R_3^3 \int_0^1 4\pi [a\rho_1(xR_1) + b\rho_2(xR_2)] x^2 dx \\ &= aR_3^3 \int_0^1 4\pi\rho_1(xR_1)x^2 dx + bR_3^3 \int_0^1 4\pi\rho_2(xR_2)x^2 dx \\ &= R_3^3 \left[a\frac{M_1}{R_1^3} + b\frac{M_2}{R_2^3} \right]. \end{aligned} \quad (4)$$

This can be put in the following form:

$$\frac{M_3}{R_3^3} = a \frac{M_1}{R_1^3} + b \frac{M_2}{R_2^3}, \quad (5)$$

and leads to:

$$R_3 = \sqrt[3]{\frac{M_3}{a \frac{M_1}{R_1^3} + b \frac{M_2}{R_2^3}}}. \quad (6)$$

Before going on to describe how the remaining acoustic variables are obtained, it is useful to discuss the relationship between the non-dimensional density profiles $\bar{\rho}(r) = \rho(r)/\rho_{\text{ref}}$, where $\rho_{\text{ref}} = M/R^3$. A few simple calculations show that:

$$\bar{\rho}_3(xR_3) = \bar{a}\bar{\rho}_1(xR_1) + \bar{b}\bar{\rho}_2(xR_2), \quad (7)$$

where

$$\bar{a} = a \frac{M_1}{R_1^3} \frac{R_3^3}{M_3}, \quad \bar{b} = b \frac{M_2}{R_2^3} \frac{R_3^3}{M_3}. \quad (8)$$

One can show that $\bar{a} + \bar{b} = 1$ by dividing Eq. 5 by M_3/R_3^3 .

The pressure can then be obtained by hydrostatic equilibrium, provided a surface value is calculated. We arbitrarily define it as follows:

$$\bar{P}_3^{\text{surf}} = a\bar{P}_1^{\text{surf}} + b\bar{P}_2^{\text{surf}} \quad (9)$$

where \bar{P}^{surf} is the non-dimensional surface values. This is calculated at x^{surf} , the minimum between x_1^{surf} and x_2^{surf} to avoid extrapolating beyond the end of either model. Using the above values of M_3 and R_3 , it is straightforward to convert \bar{P}_3^{surf} into its dimensional counterpart.

Other structural variables are also deduced from the density and Γ_1 profiles so that the resultant model is coherent from the point of view of its acoustic structure. However, no attempt is made to make this model coherent from a thermal point of view (and not all of the relevant profiles are even stored in `InterpolateModel`).

Finally, it is straightforward to generalise the above interpolation procedure to more than two models. Accordingly, `InterpolateModel` can interpolate between multiple models at a time.

3.2 The radial grid

When interpolating between multiple models, it is necessary to interpolate these models onto a common radial grid before carrying out the above calculations. In `InterpolateModel`, this common grid is an accumulation of the non-dimensional grids, $x = r/R$, from the individual models. This procedure can lead to a dense grid, especially if many models are involved in the interpolation. Duplicate points are removed if they result from coincidences between points from multiple models and are not intrinsic to a given model. As already explained above, the radial grid stops at the minimum x^{surf} value between the different models, so as to avoid extrapolating any of the models.

4 File formats

4.1 Input file for batch mode

The input file for running `InterpolateModel` in batch mode takes on the following format:

1. The first line is a header. It contains the root folder for the models involved in the interpolation, followed by the value of the gravitational constant in cgs, and optionally the input format of the models, their output format, the output format name (to allow for compatibility with `InversionPipeline` which is more limited in the file formats it can read), and an extension for the filenames of the files with the models. Default options for these last four arguments are: “FGONG”, “MOD (native endian)”, “MOD”, and “mod”. When using an argument with white spaces, these white spaces should be replaced by underscores. For example, “MOD_(native_endian)” should be used instead of “MOD (native endian)”, otherwise the arguments will be parsed incorrectly.
2. The rest of the file is composed of “blocks” which correspond to groups of models from which to produce interpolated models. These blocks can be subdivided in the following sections:
 - a one line header which contains: the number of models involved in the interpolation, the mass (in g), the radius (in cm), the luminosity (in erg/s), the metallicity, the hydrogen content, the age (in Myrs), and the temperature (in K) of the *interpolated* model. Typically, these values will have been obtained via the AIMS code and are supplied here so that `InterpolateModel` can copy them into one of the output files which can then be used by `InversionPipeline`.
 - one line per model involved in the interpolation. Each line contains an interpolation weight followed by the second part of the path to the model.
 - a blank line.

The following shows an example of an input model:

```
/home/dreese/models_inversions/Grid_mesa_MS/ 6.67428e-8 FGONG MOD_(native_endian) MOD mod
6 2.166858942431e+33 7.777566110307e+10 4.987850517228e+33 0.04774 0.60877 3.159635060727e+03 5832.44446
0.033375558977843 M1.09/LOGS_M1.09/M1.09Z0.047Y0.342/m1.09Y0.342Z0.047a1.8ovh0.2ovhe0_n25.profile.FGONG
0.892212101147946 M1.09/LOGS_M1.09/M1.09Z0.047Y0.342/m1.09Y0.342Z0.047a1.8ovh0.2ovhe0_n26.profile.FGONG
0.001448879259077 M1.09/LOGS_M1.09/M1.09Z0.057Y0.362/m1.09Y0.362Z0.057a1.8ovh0.2ovhe0_n22.profile.FGONG
0.004138929712534 M1.09/LOGS_M1.09/M1.09Z0.057Y0.362/m1.09Y0.362Z0.057a1.8ovh0.2ovhe0_n23.profile.FGONG
0.045775142612917 M1.08/LOGS_M1.08/M1.08Z0.057Y0.362/m1.08Y0.362Z0.057a1.8ovh0.2ovhe0_n24.profile.FGONG
0.023049388289682 M1.08/LOGS_M1.08/M1.08Z0.057Y0.362/m1.08Y0.362Z0.057a1.8ovh0.2ovhe0_n25.profile.FGONG

6 2.159173571070e+33 7.767911863036e+10 4.937397217440e+33 0.04944 0.60367 3.221490899622e+03 5821.25310
0.411320905014032 M1.09/LOGS_M1.09/M1.09Z0.047Y0.342/m1.09Y0.342Z0.047a1.8ovh0.2ovhe0_n26.profile.FGONG
0.133499679650532 M1.09/LOGS_M1.09/M1.09Z0.047Y0.342/m1.09Y0.342Z0.047a1.8ovh0.2ovhe0_n27.profile.FGONG
0.178146101276308 M1.08/LOGS_M1.08/M1.08Z0.047Y0.342/m1.08Y0.342Z0.047a1.8ovh0.2ovhe0_n28.profile.FGONG
0.032588686286978 M1.08/LOGS_M1.08/M1.08Z0.047Y0.342/m1.08Y0.342Z0.047a1.8ovh0.2ovhe0_n29.profile.FGONG
0.078215697547773 M1.08/LOGS_M1.08/M1.08Z0.057Y0.362/m1.08Y0.362Z0.057a1.8ovh0.2ovhe0_n24.profile.FGONG
0.166228930224377 M1.08/LOGS_M1.08/M1.08Z0.057Y0.362/m1.08Y0.362Z0.057a1.8ovh0.2ovhe0_n25.profile.FGONG
```

4.2 Output files from batch mode

When run in batch mode, `InterpolateModel` will produce a folder called `interpolated_models` and store all of the results in that folder. If the folder already exists, then `InterpolateModel` will not overwrite the folder, and will simply abort. This folder will contain the files described in the following subsections.

4.2.1 Interpolated models

These will typically take on the names `Model0.mod`, `Model1.mod`, etc. or `Model0.amdl`, `Model1.amdl`, etc., depending on the extension provided in the input file. Their format will depend on the format selected in the input file.

4.2.2 A list of interpolated models

This list can be read by `InversionPipeline` via the “Add grid” button on the Launch page. It obeys the following format:

- the first line is a header with 4 entries: the name of the grid (it will always be `Interpolated_models`), the value of the gravitational constant (in cgs), the format of the models (this will be given by output format name for `InversionPipeline`), and a URL-style prefix for the root folder with the grid of interpolated models. Accordingly, it will start with `file:///` since the files are stored locally on the computer.
- a succession of lines (one per interpolated model) with the following information:
 1. the second half of the filename of the model (*i.e.* when combined with the prefix on the first line, this produces a complete URL-style path)
 2. the mass in g,
 3. the radius in cm,
 4. the luminosity in erg/s,
 5. the initial metallicity,
 6. the initial hydrogen content,
 7. the age in Myrs,
 8. the temperature in K,
 9. and the evolutionary stage (in this case “nostage”, since the evolutionary is unknown).

The following shows the list file produced for the above input file:

```
Interpolated_models 6.67428E-8 MOD file:///home/dreese/web/java/inversion/java/InterpolateModel/interpolated_models/
Model0.mod 2.166858942431e+33 7.777566110307e+10 4.987850517228e+33 0.0477400 0.6087700 3.159635060727e+03 5832.44446 nostage
Model1.mod 2.159173571070e+33 7.767911863036e+10 4.937397217440e+33 0.0494400 0.6036700 3.221490899622e+03 5821.25310 nostage
```

NOTE: one will note though the output format was `MOD_(native_endian)`, the output format name is `MOD` as specified in the above input file. Indeed, `InversionPipeline` only reads binary mod files in native endian format, so only recognises the description `MOD`, not `MOD_(native_endian)`.

4.2.3 Summary result files

These include:

- `c2.txt` and `rho.txt`: the c^2 and ρ profiles from all of the interpolated models. The format is a simple 2 column text file with the first column being the grid, and the second the either the c^2 or ρ value. Each profile is separated by a & symbol. Such a format can easily be read by the `xmgrace` plot software.

- `c2_stat.txt` and `rho_stat.txt`: these contain statistical information on the set of c^2 and ρ profiles, namely an average profile, and the $\pm 1\sigma$ profiles. The files contain four columns with: the grid point, the average profile, the profile at -1σ , and the profile at $+1\sigma$. The underlying grid corresponds to that of the first interpolated model rather than being an accumulation of all of the grids of all of the interpolated, so as to avoid having a grid which is too dense.
- `c2_stat_adim.txt` and `rho_stat_adim.txt`: this is the same as above except that the non-dimensional profiles are used instead of the dimensional ones.

4.3 Stellar models

`InterpolateModel` accepts the following file formats for stellar models:

1. **AMDL (native endian)** or **AMDL (opposite endian)**: FORTRAN binary files generated by ASTEC
2. **FAMDL**: text (or ascii) version of the AMDL files generated by ASTEC
3. **FGONG**: exchange format under the GONG model comparison scheme, and official output format from ASTEC
4. **CESAM**: .osc text files generated by CESAM
5. **MOD (native endian)** or **MOD (opposite endian)**: FORTRAN binary file from CLES
6. **MOD (ascii)**: text file from CLES
7. **CLES**: another text file format from CLES
8. **LOSC**: a text file from LOSC but in which the model is embedded

`InterpolateModel` is also able to write files in the AMDL, FAMDL, MOD (binary) and MOD (ascii) formats. The choice of the format (for reading or saving a model) is given by a pull down menu in the load or save file dialog which pops up when clicking on one of the load/save model buttons in the **Stellar model** tab (see previous section). For the binary formats, `InterpolateModel` gives the option of selecting the endianness (i.e. the order of the individual bytes composing a number), according to the native endianness of your computer (this can be seen thanks to a label in the model page tab) or according to the opposite endianness. In general, if the user doesn't know which is the correct endianness, than trying both possibilities is the simplest solution since using the wrong endianness leads to results which are easy to recognise as wrong. `InterpolateModel` will attempt to distinguish between models generated by CESAM2k and earlier versions of CESAM by searching for "CESAM2k" in the header. If need be, `MODCONV` can be used to convert

models from one format to another. This tool is available at:

<http://www.astro.up.pt/corot/ntools/modconv/>

A description of the first four file formats can be found at:

http://www.astro.up.pt/corot/ntools/docs/CoRoT_ESTA_Files.pdf

and within the instructions to the ADIPLS pulsation code:

http://www.phys.au.dk/~jcd/adipack.n/notes/adiab_prog.ps.gz

The last four file formats will be briefly described in the following sections. This will then be followed by a table which summarises the different structural variables available in the various formats, and some useful formulas which relate these variables.

4.3.1 MOD (binary/ascii)

These contain a 5 line header (made up of 80 characters in the binary version), followed by a line which contains either 3 or 4 parameters. These are:

- the number of grid points
- the stellar radius (in cm)
- the stellar mass (in g)
- if present, the gravitational constant (in cgs)

This is then followed by a section with 6 or 7 columns (in the ascii version) or a list of entries with 5 or 6 values (in the FORTRAN binary version) which contain the following quantities:

- index of grid point (only in the ascii version)
- the radial coordinate (in cm)
- the cumulative mass (in g)
- the pressure (in cgs)
- the density (in g/cm³)
- the Γ_1 profile
- if present, an unidentified quantity (probably some sort of adiabatic discriminant – `InterpolateModel` does not make use of this column)

4.3.2 CLES format

These contain a header which ends with the expression “%%beginoscddata”. This is followed by a line which contains the following parameters:

- the stellar radius (in cm)
- the stellar mass (in g)
- the gravitational constant (in cgs)

This is followed by a line with the number of domains (in case there are double points), n_{domains} , as well as a section made up of n_{domains} lines which gives information on these domains. The following line contains the number of grid points. This is then followed by a section with 6 columns which contain the following quantities:

- the radial coordinate (in cm)
- the quantity m/r^3 (in g/cm^3)
- the pressure (in cgs)
- the density (in g/cm^3)
- the Γ_1 profile
- the quantity $-\frac{A}{r^2}$ (see Table 1)

4.3.3 LOSC format

These contain a 5 line header which is skipped. This is followed by a line with the following parameters:

- the number of grid points
- the stellar radius (in cm)
- the stellar mass (in g)
- the gravitational constant (in g)

The next three lines contain a header with column names and a list of various quantities related to the pulsation mode contained with the file. This is followed by a section with many columns. The first six columns are:

- the radial coordinate normalised by the radius
- the quantity $\eta = \frac{R^3 m}{M r^3}$
- the quantity $\frac{R}{GM} \frac{P}{\rho}$
- a normalised density, $\frac{4\pi R^3}{M} \rho$
- the Γ_1 profile
- the quantity $-\frac{A}{r^2}$ (see Table 1)

4.3.4 Comparison of different model formats

Table 1 gives the different variables from the different models and allows an easy comparison between them.

Table 1: Table which lists variables from different model formats. The first column gives the variables which are stored in `InterpolateModel`.

Variable	(F)AMDL	FGONG	CESAM	MOD	LOSC	CLES
$x = \frac{r}{R}$	$\frac{r}{R}$	r	r	r	$\frac{r}{R}$	r
Γ_1	Γ_1	Γ_1	Γ_1	Γ_1	Γ_1	Γ_1
$\frac{R^3}{M}\rho$	—	ρ	ρ	ρ	$\frac{4\pi R^3}{M}\rho$	ρ
$\frac{R^4}{M}\frac{d\rho}{dr}$	—	—	—	—	—	—
$\frac{R^4}{GM^2}P$	—	P	P	P	$\frac{R}{GM}\frac{P}{\rho}$	P
$\frac{R}{GM}c^2$	—	—	—	—	—	—
$\frac{R^3}{GM}N^2$	—	—	—	—	—	—
$\frac{m}{M}$	—	$\ln\left(\frac{m}{M}\right)$	$\ln\left(\frac{m}{M}\right)$	m	—	—
$\frac{GM}{R^2}g$	—	—	—	—	—	—
$\frac{R^2}{GM}\frac{dg}{dr}$	—	—	—	—	—	—
$\frac{R^2}{r^2}\left(1 - \frac{\rho}{\langle\rho\rangle}\right)$	—	—	—	—	—	—
$U = \frac{4\pi\rho r^3}{m}$	U	—	—	—	—	—
$A = \frac{1}{\Gamma_1}\frac{d\ln P}{d\ln r} - \frac{m}{d\ln r} = \frac{rN^2}{g}$	A	A	A	—	$-\frac{A}{r^2}$	$-\frac{A}{r^2}$
$V_g = -\frac{1}{\Gamma_1}\frac{d\ln P}{d\ln r} = \frac{Gm\rho}{\Gamma_1 pr}$	V_g	—	—	—	—	—
$\eta = \frac{R^3}{M}\frac{m}{r^3} = \frac{R^3}{GM}\frac{g}{r}$	η	—	—	—	η	$\frac{m}{r^3}$

4.3.5 Useful formulas

The following formulas can be useful for finding one structural variable from other variables:

$$c^2 = \frac{\Gamma_1 P}{\rho} \quad (10)$$

$$g = \frac{Gm}{r^2} \quad (11)$$

$$\frac{dg}{dr} = 4\pi G\rho - \frac{2Gm}{r^3} \quad (12)$$

$$\frac{R^3}{M}\rho = \frac{\eta U}{4\pi} \quad (13)$$

$$\frac{R^4}{GM}P = \frac{x^2\eta^2 U}{4\pi V_g \Gamma_1} \quad (14)$$

$$A = -r \left(\frac{g}{c^2} + \frac{1}{\rho} \frac{d\rho}{dr} \right) \quad (15)$$

$$\frac{R^3}{GM}N^2 = \frac{R^3}{GM}\frac{g}{r}A = \eta A \quad (16)$$

$$\langle\rho\rangle = \frac{m}{\frac{4}{3}\pi r^3} \quad (17)$$

4.4 GZIP compression

`InterpolateModel` can read “gzipped” text files. In order to determine whether or not a file is gzipped, `InterpolateModel` looks at the filename to see if it ends with “.gz”. The same rule also applies when saving a model from the HELAS website <http://www.astro.up.pt/helas/> onto the hard disk – choosing a filename which ends with “.gz” causes `InterpolateModel` to save a compressed file, whereas any other ending produces an uncompressed file.

5 Known bugs

Here is a list of known bugs. If you find any other, please let me know by sending me an email: (daniel.reese@obspm.fr).

- excessive zooming on plots can produce irregular behaviour

6 Copyright notices

Below is the copyright notice that goes with `InterpolateModel`.

Copyright (c) Daniel Reese 2015

This file is part of `InterpolateModel`.

`InterpolateModel` is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

`InterpolateModel` is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with `InterpolateModel`. If not, see <<http://www.gnu.org/licenses/>>.

6.1 The Dierckx.java library

This library is a java translation of a subset of the fortran library FITPACK written by Prof. P. Dierckx. The translation was carried out manually, so is not guaranteed to be bug free. The original fortran version is available at:
<http://www.netlib.org/dierckx/>

6.2 Source code for reading fortran binary files

The source code for reading fortran binary files comes from the following web-pages:
<http://docjar.com/docs/api/org/fudaa/dodico/fortran/NativeBinaryInputStream.html>
<http://docjar.com/docs/api/org/fudaa/dodico/fortran/NativeBinaryOutputStream.html>

<http://docjar.com/docs/api/org/fudaa/dodico/fortran/FortranBinaryInputStream.html>
<http://docjar.com/docs/api/org/fudaa/dodico/fortran/FortranBinaryOutputStream.html>
and are covered by the GNU GPL2 License. They have been corrected and modified so as to meet the needs of InterpolateModel.

6.3 Supplementary notices

Some of the code comes from other sources. The corresponding copyright notices are reproduced below:

Notice number 1

@(#)OptionPaneDemo.java 1.9 04/07/26

Copyright (c) 2004 Sun Microsystems, Inc. All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. ("SUN") AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that this software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

Notice number 2

Copyright (c) Ian F. Darwin, <http://www.darwinsys.com/>, 1996-2002.
All rights reserved. Software written by Ian F. Darwin and others.
\$Id: LICENSE,v 1.8 2004/02/09 03:33:38 ian Exp \$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Java, the Duke mascot, and all variants of Sun’s Java "steaming coffee cup" logo are trademarks of Sun Microsystems. Sun’s, and James Gosling’s, pioneering role in inventing and promulgating (and standardizing) the Java language and environment is gratefully acknowledged.

The pioneering role of Dennis Ritchie and Bjarne Stroustrup, of AT&T, for inventing predecessor languages C and C++ is also gratefully acknowledged.