# InversionKit

Version 1.4

Daniel Reese
and
Sergei Zharkov

August 2011

# Contents

**Acknowledgements**

This program was written by Daniel Reese and Sergei Zharkov,whose work is supported by the European Helio- and Asteroseismology Network (HELAS), a major international collaboration funded by the European Commission's Sixth Framework Programme.

# 1 Getting started

## 1.1 Running the program

`InversionKit` runs under Java 5.0 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 the file `InversionKit.jar` from the HELAS website:

`http://helas.group.shef.ac.uk/science/inversions/InversionKit/index.php`

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

```
java -jar InversionKit.jar
```

If you are planning to do calculations involving large data and kernels sets, you may need to allocate a larger amount of memory to run the program. To allocate, for example, 500 MB of memory (rather than the default 64 MB), use the following command:

```
java -Xmx500m -jar InversionKit.jar
```

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

## 1.2 Using the program

Once `InversionKit` is running, the user has several options:

- calculate theoretical frequency shifts or rotational splittings from target profiles
- invert frequency shifts or rotational splittings to find structural or rotational profiles
- estimate the stellar mean density through SOLA inversions and through scaling laws
- combine the above options

In order to do the above, the user must first of all:

1. load a stellar model
2. load a set of eigenmodes, which enables the program to calculate a set of kernels
3. load a set of frequency shifts or rotational splittings

The above operations are done in different tabs within the program. These tabs are:

- `Rotational inversion`: this tab inverts for the rotation profile and also allows the user to load a target rotation profile

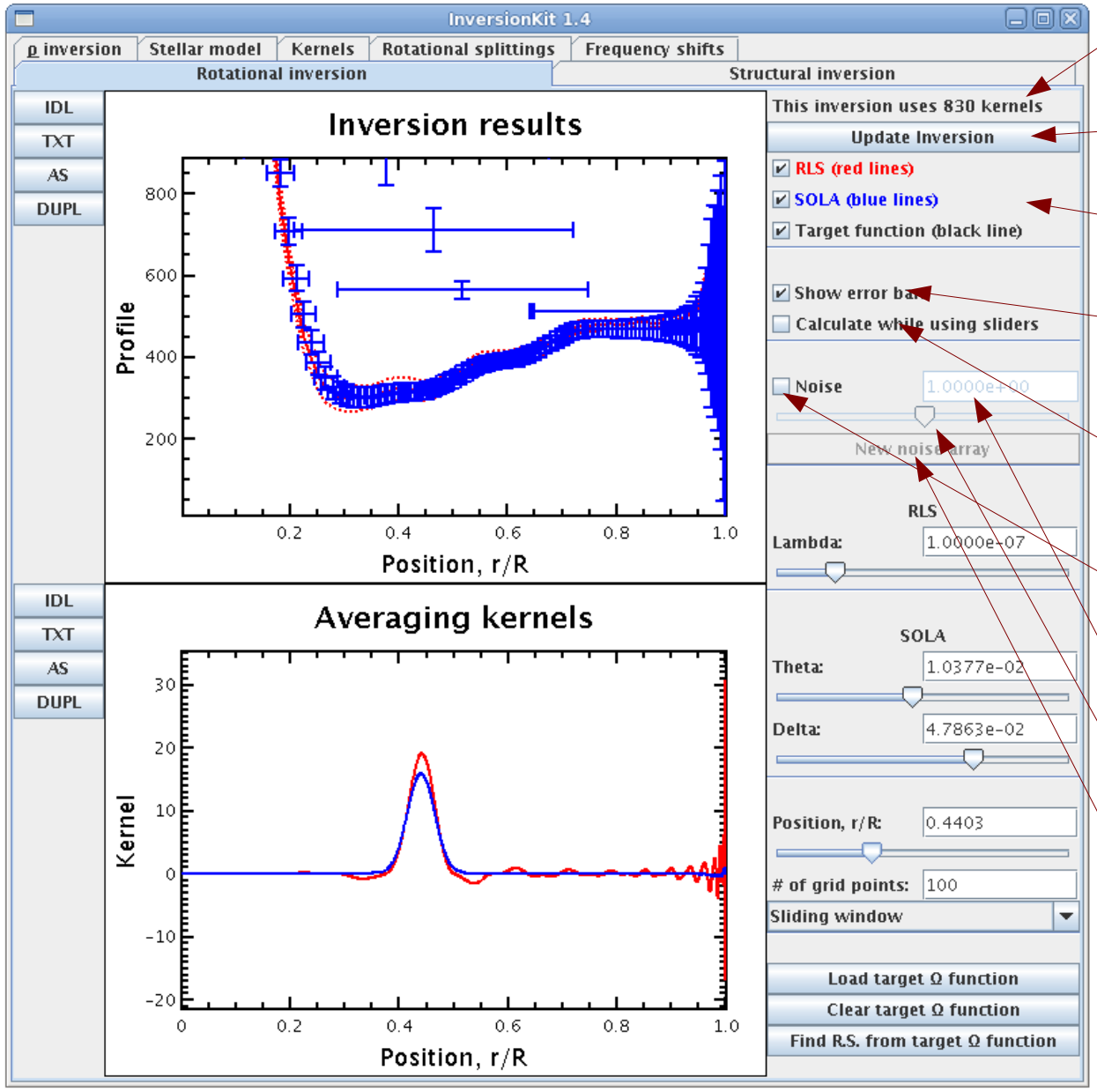
- `Structural inversion`: this tab does structural inversions on pairs of structural profiles and allows the user to load target structural profiles.
- $\underline{\rho}$ `inversion`: this tab allows the user to estimate the mean density of the star using a SOLA type structural inversion and two different scaling laws
- `Stellar model`: this tab allows the user to load or generate a stellar model
- `Kernels`: this tab allows the user to load eigenfunctions and calculate the corresponding rotational and structural kernels
- `Rotational splittings`: this tab allows the user to load/generate/edit rotational splittings data
- `Frequency shifts`: this tab allows the user to load/generate/edit frequency shifts data

The next few pages gives a brief description of the different buttons and options which appear in these tabs. The `Structural Inversion` tab is similar to the `Rotational Inversion` tab; therefore only extra features are described. The `Rotational Splittings` tab is omitted altogether because it is quite similar to the `Frequency Shifts` tab. Moreover, the actual appearance of these tabs may vary from one platform to another depending on the Java installation.

## 1.3 Preliminary remarks

Some of the operations can take some time. For example, loading 800 eigenmodes and calculating the corresponding kernels can take typically 40s. Doing a rotational inversion with 800 kernels takes typically 7.5s. When the program is calculating, it is best just to wait and let it finish what it is doing.

Also, unlike previous versions of `InversionKit`, most of the the check boxes, sliders and text fields in the `Rotation inversion`, `Structural inversion` and $\underline{\rho}$ `inversion` tabs will only partially update the inversion, i.e. if the model, the kernels, frequency shift data or rotational splittings have been modified, this will not be taken into account. To take these modifications into account, please use the `Update Inversion` buttons. Besides the `Update Inversion` buttons, the other buttons which cause the inversion to be fully updated are: the `Number of grid points` text field, the `Integration method` combo box (see Sect. 3.4), and the `Kernel type` combo box.




this says how many kernels are being used in the current inversion

this updates/refreshes the inversion

these determine which functions are shown

this determines whether or not to show the error bars in both types of inversion

when selected, the program tries to calculate the inversion at each intermediate positions on the sliders below

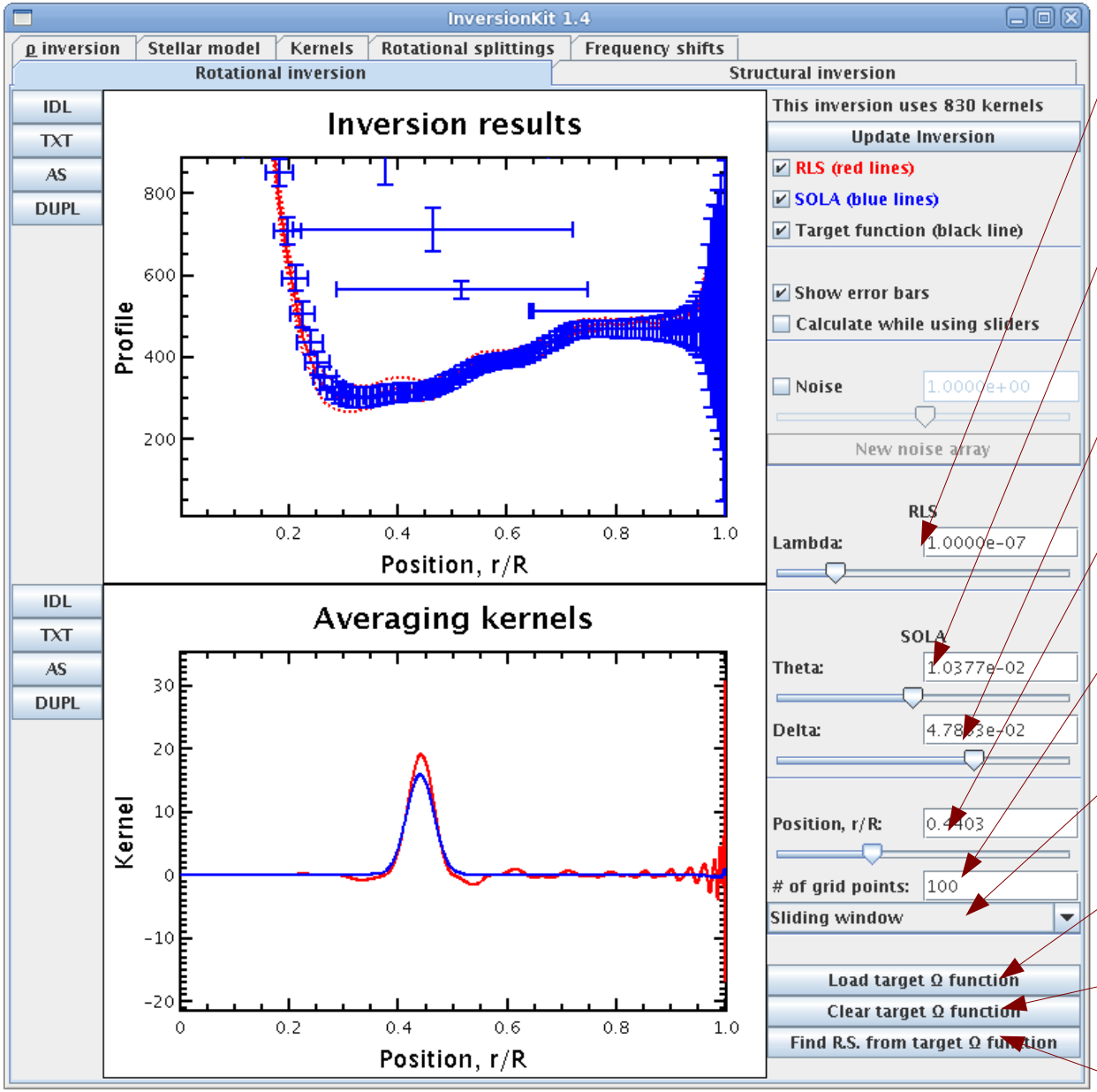
when selected, artificial noise is added to the data before the inversion

text field which determines the amplitude* of the noise

slider which determines the amplitude* of the noise

this generates a new realisation of noise

*In InversionKit 1.4, the 1-σ noise amplitude is given by the product of the value in the noise text field, and the 1-σ error bar for each measurement given in the Rotational splittings tab. The same applies to the other inversions.




text field and slider which determine the amount of regularisation in an RLS inversion (see Formula section)

text field and slider which determine the amount of regularisation in a SOLA inversion (see Formula section)

text field and slider which determine the widths of the averaging kernels in a SOLA inversion (see Formula section)

text field and slider which determine the target grid point for which to show the averaging kernels in either type of inversion

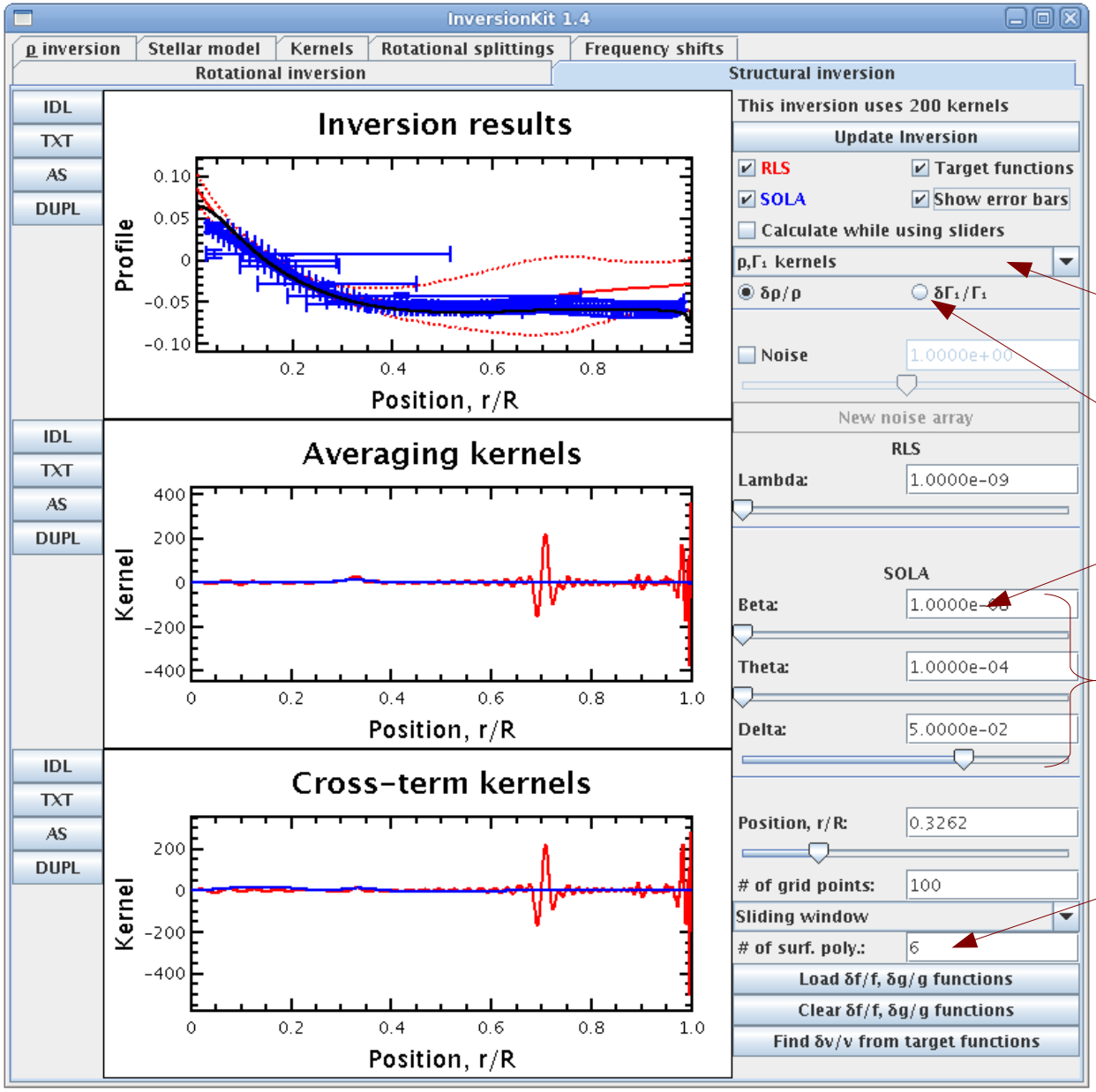
text field which determines the number of grid points to be used in the inversions

this determines whether to use a sliding window or Chebyshev polynomial approach (see Integration Method section)

this load a target rotation function (see File format section)

this clears the current target function

this calculates theoretical rotational splittings from the target function



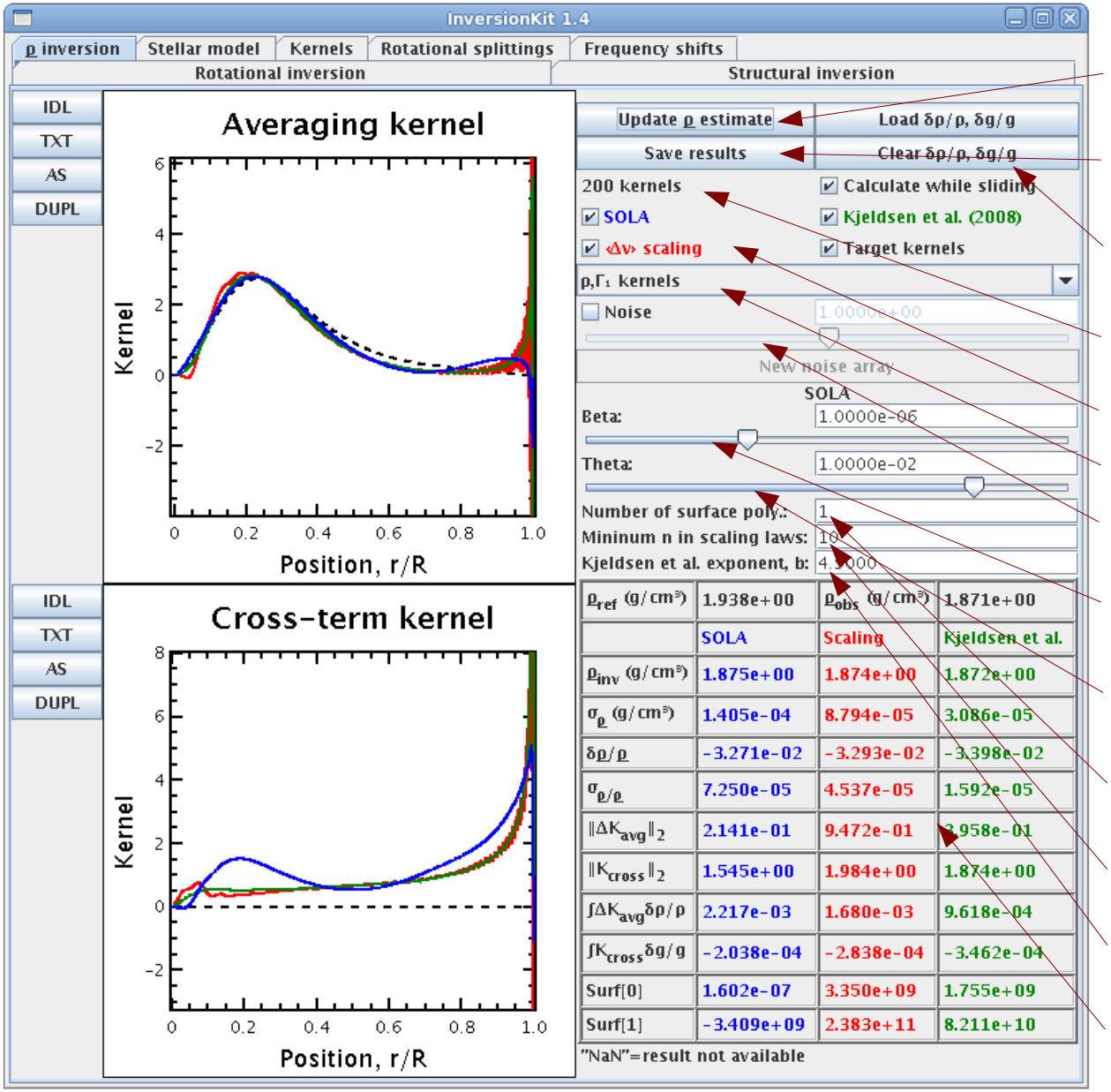

combo box which gives the choice between different types of structural kernels

this determines which of two structural functions will be shown

text field and slider which determine the ratio between producing “nice” averaging kernels and reducing cross-term kernels (see Formula section)

these sliders and text fields can be adjusted independently for the two different structural inversions

text field which determines the number of legendre polynomials used in ad-hoc modeling of surface effects (the polynomial degrees go from 0 to this value minus 1)




this updates/refreshes the mean density estimate

saves mean density estimates and associated results to a file

load/clear structural profiles (idem. on Structural Inversion tab)

the number of kernels used in the mean density estimate

selects which functions to display

selects which structural pair to use

regulates optional noise

regulates between optimising $K_{avg}$ and minimising $K_{cross}$

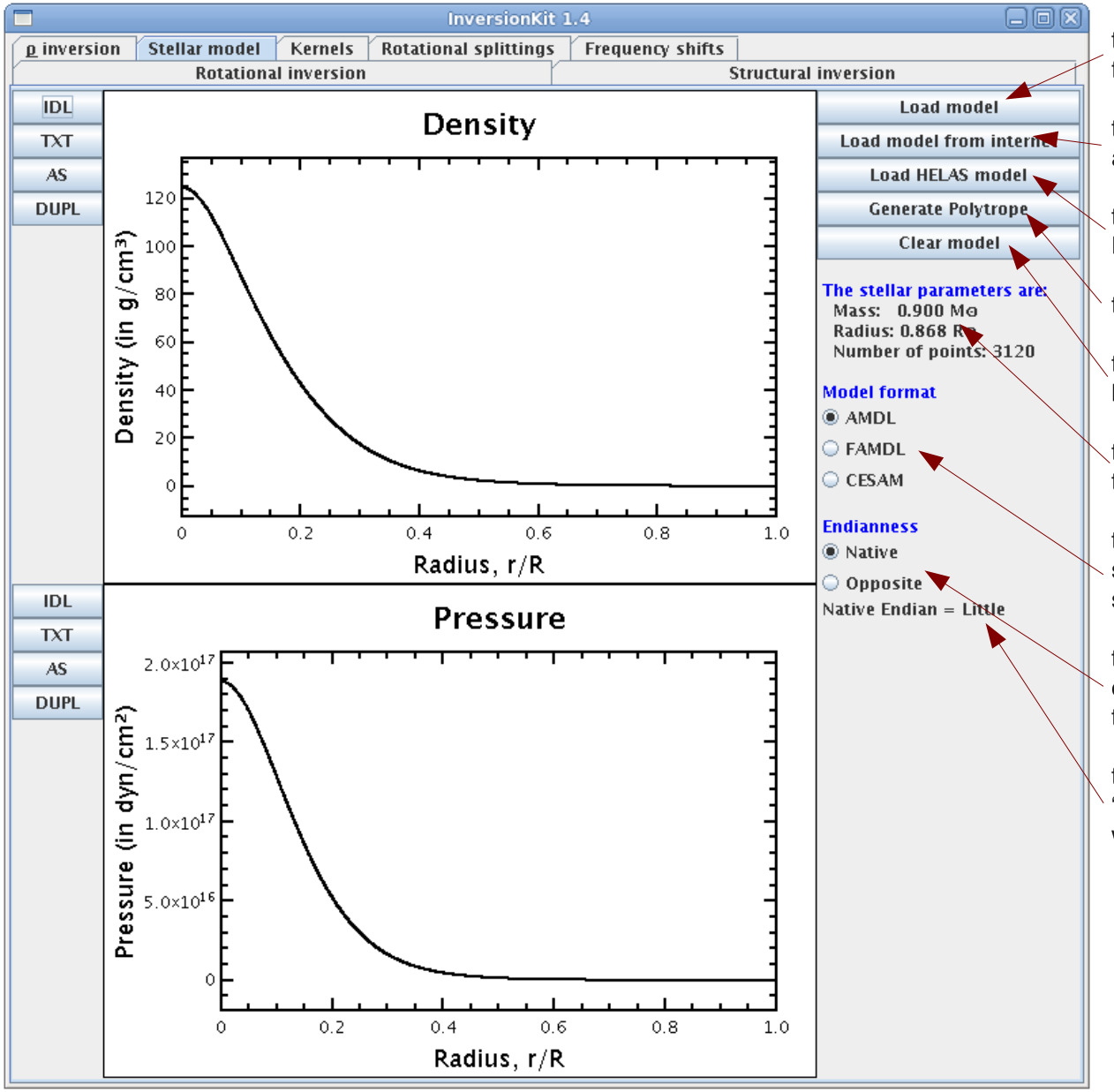
determines the amount of regularisation in SOLA inversion

number of polynomials used to model surface effects

minimum radial order used in scaling and Kjeldsen's law

exponent in Kjeldsen's law

table with different resuls for SOLA inversion, scaling and Kjeldsen's law (see text for details)




this loads a stellar model from a file

this loads a stellar model from a specified URL

this loads a model from the HELAS website

this generates a polytropic model

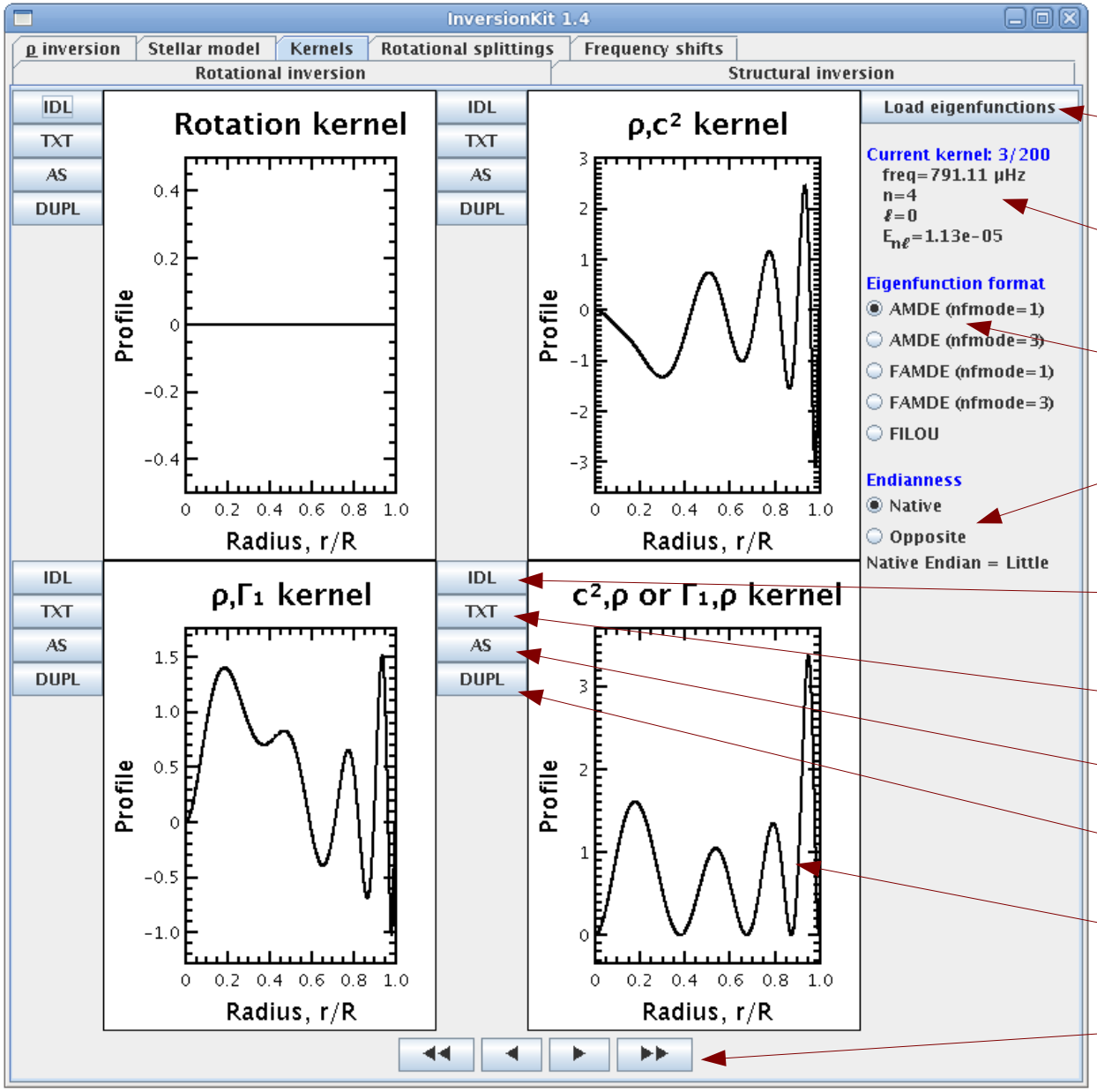
this removes the currently loaded model

this gives some information on the model currently loaded

this specifies the format of the stellar model (see Format section)

this specifies the "endianness" of a binary file (only applies to the AMDL format)

this specifies the native "endianness" of the machine on which InversionKit is running




this loads eigenfunctions and calculates all of the kernels from these

this gives information on the different eigenmodes/ kernels

this specifies the format of the file with the eigenfunctions (see File Format section)

this specifies the "endianness" of the eigenfunction file (only applies to AMDE format)

this produces a file with IDL instructions to reproduce the plot

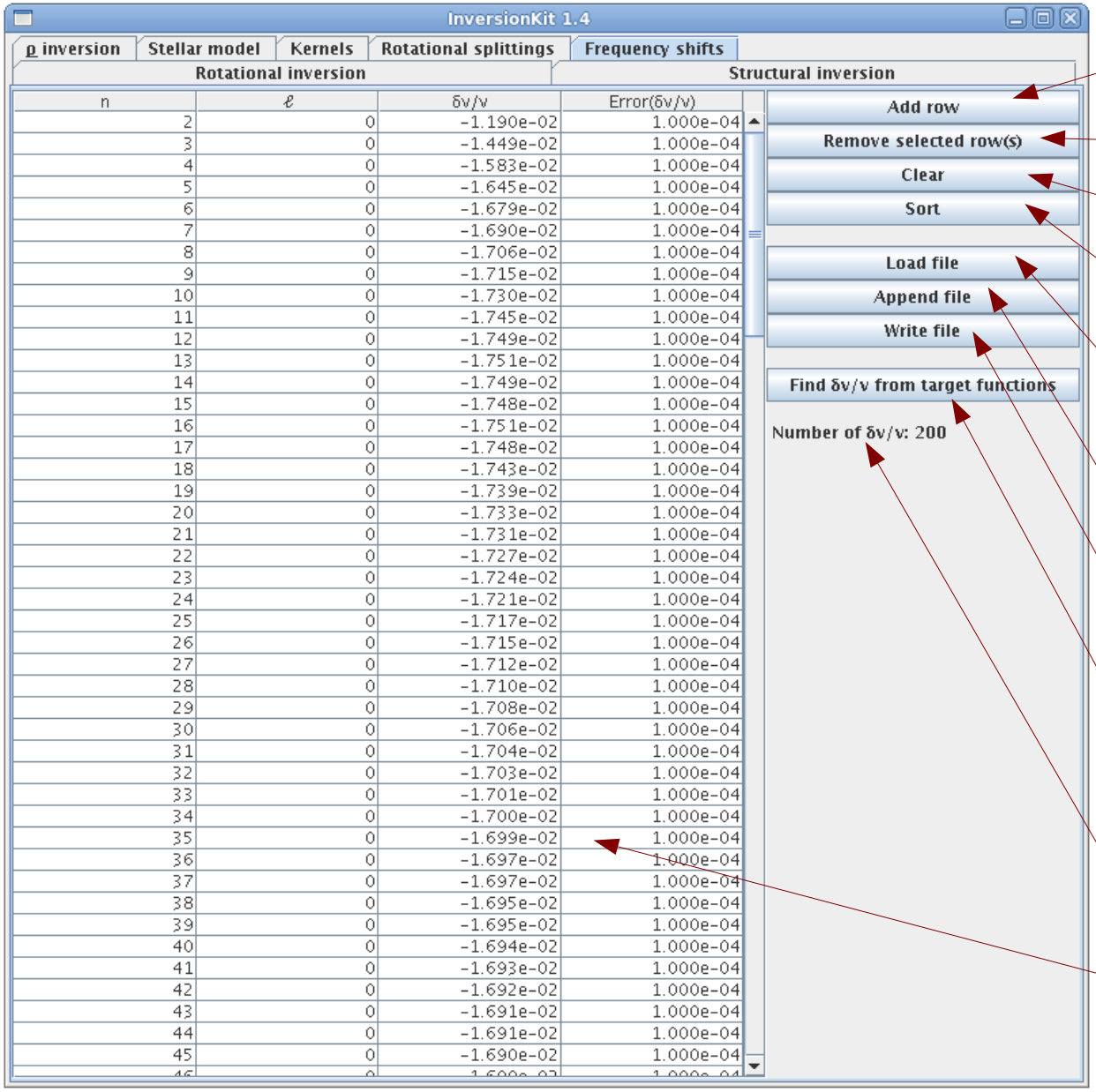
this produces a txt file with the data from the plot

this is the auto-scale button

this duplicates the plot into a new window

click and drag on the plot area to zoom in

these are navigation buttons so as to look at the different kernels that are loaded


InversionKit 1.4
ρ inversion
Stellar model
Kernels
Rotational splittings
Frequency shifts
Rotational inversion
Structural inversion
n
ℓ
δν/ν
Error(δν/ν)
Add row
Remove selected row(s)
Clear
Sort
Load file
Append file
Write file
Find δν/ν from target functions
Number of δν/ν: 200
this adds a blank row at the end of the list
this removes the selected rows
this clears all of the data
this sorts the data according to (l,n)
this opens a file and replaces the current data with the file's data (see File Format section)
this opens a file and appends its data to the end of the current data
this writes a file with the current data
this calculates theoretical frequency shifts from the target structural functions loaded in the structural inversion or the mean density inversion tabs
this gives the number of frequency shift data currently loaded
the entries in this table can be edited

# 2 File formats

## 2.1 Stellar models

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

1. **AMDL**: FORTRAN binary files generated by ASTEC
2. **FAMDL**: text files generated by ASTEC
3. **OSC**: text files generated by CESAM

A description of these 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 choice of the format is given by a menu in the `Model Page` tab (see previous section). `InversionKit` 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 convert models from one format to another. This tool is available at:

`http://www.astro.up.pt/corot/ntools/modconv/`

## 2.2 Eigenmodes

`InversionKit` accepts eigenmodes in one of the following formats:

- **AMDE**: FORTRAN binary files produced by ADIPLS
- **"FAMDE"**: text version of the AMDE format
- **FILOU**: text files produced by FILOU

### 2.2.1 The AMDE format

The AMDE format is a FORTRAN binary output file from ADIPLS. `InversionKit` accepts AMDE files produced with the options `nfmode=1` or `nfmode=3`. For more details on this format, please consult the instructions to the ADIPLS code:

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

### 2.2.2 The FAMDE format

The FAMDE format is a text version of the AMDE format produced by ADIPLS (with `nfmode=1` or `nfmode=3`) plus an additional header as described in the 1996 inversion workbench description. At the end of the header is a succession of 3 {`CTRL+L, CTRL+J`} in a row, which separates it from the rest of the file. `InversionKit` skips the header and reads data which comes after. The format obeys the following rules:

- Files are divided into different sections. These are given by the following lists, as indicated by the semi-colons:
  - **nfmode = 1**: `gp` array; `nr`; `x` and `y` arrays; `gp` array; `nr`; `x` and `y` arrays; etc.
  - **nfmode = 3**: `nr`; `x` array; `gp` array; `y` array; `gp` array; `y` array; etc.
- where:
  - `nr` = radial resolution
  - `x` = radial grid (size: `nr`)
  - `gp` = global parameters for a given mode (size: 50)
  - `y` = array with eigenfunctions (size: 6×`nr` for `nfmode=1`, and 2×`nr` for `nfmode=3`)
- Each section starts on a new line. The number of entries per line does not matter, but each entry needs to be separated by at least one space, comma or tab.
- Blank lines are not allowed, except in the header.

An auxiliary Java tool, `ReadFAMDE`, which reads, combines and converts between AMDE and FAMDE files, is available at:

`http://helas.group.shef.ac.uk/science/pulsations/ReadFAMDE/index.php`

### 2.2.3 The FILOU format

The FILOU format can be described as a series of individual eigenmodes which are defined by a header and a table. The header contains a number of key parameters, preceded by descriptive character strings. The relevant parameters are:

- the harmonic degree $\ell$ (preceded by "DEGRE DU MODE L :")
- the number of grid points (preceded by "Nombre de points du reseau du modele :")
- the normalised squared frequency (preceded by "Frequence carree normalisee =")
- the radial order (preceded by "noeuds =")
- the frequency in $\mu$Hz (preceded by "Frequence en micro Hz =")

Three supplementary lines appear between the frequency in $\mu$Hz and the start of the table. The table contains 5 columns, each of which are 15 characters wide. The first column is the normalised radial position and the remaining 4 give the variables $y_{01}$ .. $y_{04}$ which are defined in Suárez and Goupil, 2008 (Astrophys. Space Sci. 316, 155-161). The

two first functions $y_{01}$ and $y_{02}$ are used to calculate normalised displacements which can then be used to find the different kernels. Although the FILOU oscillation code is able to take into account the effects of rotation using perturbation theory, `InversionKit` is only set up to calculate and do inversions using eigenmodes and eigenfrequencies from non-rotating models.

## 2.3 Target profiles

The files which contain the target profile(s) need to obey the following rules:

- On a given line, anything following a "#" is treated like a comment and ignored.
- There are 2 columns for the target rotation profile and 3 columns for the target structural profiles.
- The first column corresponds to the underlying grid. This grid needs to be in strictly ascending order. If the span covered by the grid is smaller than the span covered by the eigenmode/kernel grid, then the target profile(s) will be extrapolated to fit the eigenmode/kernel grid.
- The next column(s) contain the target profile(s).
- A line with the wrong number of entries (after removal of comments) are discarded but provoke a warning message (except if there are no entries).

*Note:*

- the underlying grid for the target profiles needs to be non-dimensionalised (*i.e.* the grid corresponds to $x = r/R$, where $R$ is the radius of the model)
- the same target profiles are used both for structural inversions and for mean density estimates.

## 2.4 Frequency shifts or rotational splittings

The files which contain frequency shifts or rotational splittings need to obey the following rules:

- On a given line, anything following a "#" is treated like a comment and ignored.
- There are 4 columns. These are:
  1. Integer entry which corresponds to the radial order $n$.
  2. Integer entry which corresponds to the harmonic degree (or order) $\ell$.
  3. Floating number which corresponds to either the frequency shift ($\delta\nu/\nu$) or the rotational splitting (R.S.)
  4. Floating number which corresponds to the error on either the frequency shift or the rotational splitting.
- A line with the wrong number of entries (after removal of comments) are discarded but provoke a warning message (except if there are no entries).

## 2.5 GZIP compression

`InversionKit` can read "gzipped" text files. In order to determine whether or not a file is gzipped, `InversionKit` 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 `InversionKit` to save a compressed file, whereas any other ending produces an uncompressed file.

# 3 Formulas

## 3.1 Rotation inversions

### 3.1.1 Description of the problem

A stellar rotation profile which only depends on depth, $\Omega(r)$, shifts the frequencies by an amount that is proportional to $m$, the azimuthal order:

$$R_{n\ell} \equiv \frac{\nu_{n\ell m} - \nu_{n\ell 0}}{m} = \int_0^R K_\Omega^{n\ell}(r)\Omega(r)dr \tag{1}$$

where $n$ is the radial order, $\ell$ the harmonic degree and $K_\Omega^{n\ell}$ the rotational kernel. The quantity $R_{n\ell}$ is the rotational splitting and is deduced from observations for a set of $(n, \ell)$ pairs. The associated rotational kernels can be calculated from the corresponding pulsation modes in a non-rotating reference model (see Section 3.1.2 for an explicit expression). The goal of an inversion procedure is then to deduce the unknown rotation profile, $\Omega(r)$ from $R_{n\ell}$ and $K_\Omega^{n\ell}$. Two methods for doing this are implemented in `InversionKit`: Regularised Least Squares (RLS) and Subtractive Optimally Localised Averages (SOLA). These are described in Sections 3.1.3 and 3.1.4, respectively.

### 3.1.2 Expression for the rotational kernel

The rotational kernel can be expressed as follows:

$$K_\Omega^{n\ell} = \frac{\rho_0 r^2}{I}\left(\xi^2 + \ell(\ell+1)\eta^2 - 2\xi\eta - \eta^2\right) \tag{2}$$

where

$$\begin{aligned}
\vec{\xi} &= \xi Y_m^\ell \vec{e}_r + \eta\left(\frac{\partial Y_m^\ell}{\partial\theta}\vec{e}_\theta + \frac{1}{\sin\theta}\frac{\partial Y_m^\ell}{\partial\varphi}\vec{e}_\phi\right) \\
&= \text{the Lagrangian displacement resulting from the pulsation} \\
I &= \int_{r=0}^R \rho_0(r)\left(\xi^2 + \ell(\ell+1)\eta^2\right) r^2 dr \\
\rho_0 &= \text{the equilibrium density}
\end{aligned}$$

It is important to note that the eigenfunctions $\xi$ and $\eta$ only depend on $(n, \ell)$ and not $m$ in a non-rotating reference model.

### 3.1.3 RLS – Regularised Least Squares

Both RLS and SOLA inversions involve minimising "cost" functions which will be represented by the letter $J$ in what follows. In an RLS inversion, $J$ is defined as:

$$J(f) = \sum_{l=1}^{L} \left\{ \frac{R_l - \int_0^R K_\Omega^l(r) f(r) dr}{\sigma_l} \right\}^2 + \Lambda \left\langle \frac{1}{\sigma^2} \right\rangle \int_0^R \left\{ \frac{d^2 f}{dr^2} \right\}^2 dr \tag{3}$$

where $R_l$ is the rotational splittings, $\sigma_l$ the corresponding errors, $K_\Omega^l$ the corresponding rotational kernels, and $\left\langle \frac{1}{\sigma^2} \right\rangle = \frac{1}{L} \left( \sum_{l=1}^{L} \frac{1}{\sigma_l^2} \right)$. The index $l$ is used to represent the pairs $(n, \ell)$, and $L$ the number of such pairs. $\Lambda$ is a trade-off parameter between conforming to data and regularising the solution, and can be regulated in the `Rotational inversion` tab. The function $f$ that minimises $J$ is the inversion result and corresponds to the profile which best reproduces the rotational splittings while satisfying the regularisation constraint.

### 3.1.4 SOLA – Subtractive Optimally Localised Averages

The SOLA inversion procedure is one of several inversion methods which focuses on constructing "nice" averaging kernels $K_{\text{avg.}}$ (see Section 3.1.5 for an explanation on averaging kernels). It was first introduced by Pijpers & Thompson (1992, A&A 262, L33) and has the advantage of being less computationally expensive than other OLA inversions. In what follows, a description of the SOLA method is given.

For a given grid point, $r_0$, a target function $T(r_0, r)$ is chosen. The cost function, $J$, is then set up so as to minimise the difference between the averaging kernel and $T(r_0, r)$ while reducing the effects of the observational errors $\sigma_l$.

$$\begin{aligned} J(c_l(r_0)) &= \int_0^R \left\{ T(r_0, r) - K_{\text{avg.}}(r_0, r) \right\}^2 dr + \frac{\tan\theta \sum_{l=1}^{L} \left( c_l(r_0) \sigma_l \right)^2}{\langle \sigma^2 \rangle} \\ &+ \lambda \left\{ 1 - \int_0^R K_{\text{avg.}}(r_0, r) dr \right\} \end{aligned} \tag{4}$$

where

$$\begin{aligned} c_l(r_0) &= \text{coefficients from the inversion} \\ \langle \sigma^2 \rangle &= \frac{1}{L} \sum_{l=1}^{L} \sigma_l^2 \\ \theta &= \text{a trade-off parameter between optimising } K_{\text{avg.}} \text{ and reducing the error,} \\ &\quad \text{and which can be adjusted by the user} \\ T(r_0, r) &= \text{a target function} \\ K_{\text{avg.}}(r_0, r) &= \sum_{l=1}^{L} c_l(r_0) K_\Omega^l(r) = \text{the averaging kernel} \\ \lambda &= \text{Lagrangian multiplier used to ensure } \int_0^R K_{\text{avg.}}(r_0, r) dr = 1 \end{aligned}$$

In `InversionKit`, the following form has been chosen for the target function:

$$T(r_0, r) = Ar \exp\left\{ -\left[ \frac{r - r_0}{\Delta \cdot c(r_0)} + \frac{\Delta \cdot c(r_0)}{2r_0} \right]^2 \right\} \tag{5}$$

where $A$ is a normalisation factor such that $\int_0^R T(r_0, r)dr = 1$. This type of target function was taken from Rabello-Soares et al. (1999, MNRAS 309, 35), has a maximum at $r_0$, and behaves like a Gaussian, except that it goes to zero at $r = 0$. The width is $\Delta \cdot c(r_0)$, where $c(r_0)$ is the sound velocity at $r_0$ (and shouldn't be confused with the inversion coefficients $c_l(r_0)$) and $\Delta$ a free parameter which can be adjusted by the user. This form for the width reflects the resolving power of the kernels (Thompson 1993, ASPCS 42, 141). The solution $\Omega_{\text{inv.}}$ is then constructed as follows:

$$\Omega_{\text{inv.}}(r_0) = \sum_{l=1}^{L} c_l(r_0) R_l \tag{6}$$

for a set of grid points $r_0$.

### 3.1.5 Error bars and averaging kernels

In linear inversion procedures, including RLS and SOLA, the inverted rotation profile is a linear combination of the rotational splittings:

$$\Omega_{\text{inv.}}(r_0) = \sum_{l=1}^{L} c_l(r_0) R_l \tag{7}$$

where $c_l(r_0)$ represents the inversion coefficients. Applying this linear combination to Eq. (1) yields the following relation:

$$\Omega_{\text{inv.}}(r_0) = \int_0^R \sum_{l=1}^{L} c_l(r_0) K_\Omega^l(r) \Omega(r) dr = \int_0^R K_{\text{avg.}}(r) \Omega(r) dr \tag{8}$$

From this, we see that $\Omega_{\text{inv.}}(r_0)$ is actually an average of the true rotation profile $\Omega(r)$, in which $K_{\text{avg.}}$ plays the role of a weight function, hence the name "averaging kernel". Such functions are useful for assessing the quality of the inversion (see Christensen-Dalsgaard et al., 1990, MNRAS 242, 353). `InversionKit` allows the user to plot the averaging kernel from both types of inversion methods for the different grid points, $r_0$, used in the inversion.

Additionally, $K_{\text{avg.}}$ is used to calculate the horizontal error bars in SOLA inversions: the left end of the error bar corresponds to the first quartile point, the right end is the third quartile point and the point $r_0$ is redefined to be the second quartile point. These quartile points are calculated through integration starting from the centre of the star.

From the inversion coefficients, $c_l(r_0)$, and the 1-sigma measurement errors, $\sigma_l$, it is possible to define a 1-sigma error on the inversion result at each point, $r_0$:

$$\mathcal{E}(r_0) = \sqrt{\sum_{l=1}^{L} \left( c_l(r_0) \sigma_l \right)^2} \tag{9}$$

$\mathcal{E}(r_0)$ is then used to draw the vertical error bars on both RLS and SOLA inversions. For RLS inversions, this is represented by dotted red lines below and above the inverted profile, whereas for SOLA inversions, vertical error bars are plotted.

**Note:** For limited sets of modes, the averaging kernels can be highly oscillatory, meaning that multiple values could qualify as first, second and third quartile points. This makes the above definition for the horizontal error bars essentially meaningless. In such a situation, `InversionKit` arbitrarily uses the innermost quartile points, which of course introduces a bias towards the centre of the star. It is therefore important to look at the averaging kernels, rather than simply trusting the horizontal error bars.

## 3.2 Structural inversions

### 3.2.1 Description of the problem

A structural inversion problem typically takes the following form:

$$S_{n\ell} \equiv \frac{\delta\nu_{n\ell}}{\nu_{n\ell}} = \int_0^R K^{n\ell}_{a,b}(r)\frac{\delta a}{a}dr + \int_0^R K^{n\ell}_{b,a}(r)\frac{\delta b}{b}dr + \frac{F_{\text{surf.}}(\nu_{n\ell})}{E_{n\ell}} \tag{10}$$

where $S_{n\ell}$ is the relative frequency difference (or shift) between the observed and calculated frequencies (obtained for $m = 0$), $a$ and $b$ structural profiles (such as $c_0^2$, $\rho_0$, $\Gamma_1$ etc.), $\delta a$ and $\delta b$ the differences between the true structural profiles and the ones from the reference model, $F_{\text{surf.}}$ a slowly varying function which represents unknown surface effects, and $E_{n\ell}$ the mode inertia:

$$E_{n\ell} = \frac{\int_0^R \left[\xi^2 + \ell(\ell+1)\eta^2\right]\rho r^2 dr}{M\left[\xi(R_{\text{phot.}})^2 + \ell(\ell+1)\eta(R_{\text{phot.}})^2\right]} \tag{11}$$

$R$ and $R_{\text{phot.}}$ are the surface and photospheric radii of the model. The goal of a structural inversion procedure is to find the unknown functions $\delta a/a$ and $\delta b/b$ from a set of relative frequency shifts, $S_{n\ell}$, and from the associated structural kernels, $K^{n\ell}_{a,b}$ and $K^{n\ell}_{b,a}$ (which are calculated from the pulsation modes of the model). Although similar to the rotational inversion problem described in Section 3.1.1, there are some noteworthy differences. Firstly, the inversion procedure needs to find two unknown functions – this raises the possibility of cross-talk between the two solutions. Secondly, there is an additional surface term which needs to be suppressed.

In what follows, explicit expressions are given for structural kernels associated with the structural pairs $(c_0^2, \rho_0)$ and $(\Gamma_1, \rho_0)$. These expressions are obtained by perturbing the variational expression for the frequency, then using the the variational principle to remove terms related to the variation of the eigenfunctions. The following expressions are from Gough & Thompson (1991).

### 3.2.2 $c^2$, $\rho$ kernels

The relative frequency variations are given by

$$\frac{\delta\nu}{\nu} = \int_{r=0}^R \left[K_{c^2,\rho}(r)\frac{\delta c_0^2(r)}{c_0^2(r)} + K_{\rho,c^2}(r)\frac{\delta\rho_0(r)}{\rho_0(r)}\right]dr \tag{12}$$

in which the kernels take on the following expressions:

$$K_{c^2,\rho} = \frac{\rho_0 c_0^2 \chi^2 r^2}{2I\omega^2} \tag{13}$$

$$K_{\rho,c^2} = \frac{\rho_0 r^2}{2I\omega^2}\left\{c_0^2\chi^2 - \omega^2\left(\xi^2 + \ell(\ell+1)\eta^2\right) - 2g_0\xi\chi - 4\pi G\int_{s=r}^{R}\left(2\rho_0\xi\chi + \frac{\mathrm{d}\rho_0}{\mathrm{d}s}\xi^2\right)ds \right.$$
$$\left. +2g_0\xi\frac{\mathrm{d}\xi}{\mathrm{d}r} + 4\pi G\rho_0\xi^2 + 2\left(\xi\frac{\mathrm{d}\psi}{\mathrm{d}r} + \frac{\ell(\ell+1)\eta\psi}{r}\right)\right\} \tag{14}$$

where

$$\begin{aligned}
\omega &= 2\pi\nu \\
\chi &= \frac{\vec{\nabla}\cdot\vec{\xi}}{Y_m^\ell} = \frac{\mathrm{d}\xi}{\mathrm{d}r} + \frac{2\xi}{r} - \frac{\ell(\ell+1)\eta}{r} \\
\rho &= -\frac{\mathrm{d}\rho_0}{\mathrm{d}r}\xi - \rho_0\chi \\
\psi &= -\frac{4\pi G}{2\ell+1}\left[\int_{s=0}^{r}\rho(s)\frac{s^{\ell+2}}{r^{\ell+1}}ds + \int_{s=r}^{R}\rho(s)\frac{r^\ell}{s^{\ell-1}}ds\right] \\
\frac{\mathrm{d}\psi}{\mathrm{d}r} &= -\frac{4\pi G}{2\ell+1}\left[-(\ell+1)\int_{s=0}^{r}\rho(s)\frac{s^{\ell+2}}{r^{\ell+2}}ds + \ell\int_{s=r}^{R}\rho(s)\frac{r^{\ell-1}}{s^{\ell-1}}ds\right] \\
m_0 &= 4\pi\int_{s=0}^{r}\rho_0(s)s^2ds \\
g_0 &= \frac{Gm_0}{r^2}
\end{aligned}$$

and quantities with the subscript "0" refer to the equilibrium model.

### 3.2.3 $\Gamma_1$, $\rho$ kernels

The relative frequency variations are given by

$$\frac{\delta\nu}{\nu} = \int_0^R\left[K_{\Gamma_1,\rho}(r)\frac{\delta\Gamma_1(r)}{\Gamma_1(r)} + K_{\rho,\Gamma_1}(r)\frac{\delta\rho_0(r)}{\rho_0(r)}\right]dr \tag{15}$$

Expressions for these kernels can be deduced from the previous set of kernels:

$$K_{\Gamma_1,\rho} = K_{c^2,\rho} = \frac{\rho_0 c_0^2\chi^2 r^2}{2I\omega^2} \tag{16}$$

$$\begin{aligned}
K_{\rho,\Gamma_1} &= K_{\rho,c^2} - K_{c^2,\rho} + \frac{Gm\rho_0}{r^2}\int_{s=0}^{r}\frac{K_{c^2,\rho}(s)}{p_0(s)}ds + \rho_0 r^2\int_{s=r}^{R}\frac{4\pi G\rho_0}{s^2}\left(\int_{t=0}^{s}\frac{K_{c^2,\rho}(t)}{p_0(t)}dt\right)ds \\
&= K_{\rho,c^2} - K_{c^2,\rho} + \frac{Gm\rho_0}{r^2}\int_{s=0}^{r}\frac{\Gamma_1\chi^2 s^2}{2I\omega^2}ds + \rho_0 r^2\int_{s=r}^{R}\frac{4\pi G\rho_0}{s^2}\left(\int_{t=0}^{s}\frac{\Gamma_1\chi^2 t^2}{2I\omega^2}dt\right)ds
\end{aligned} \tag{17}$$

### 3.2.4 RLS

The cost function, $J$, is defined as:

$$\begin{aligned}
J(f,g,a_n) &= \sum_{l=1}^{L}\left\{\frac{S_l - \int_0^R K_{a,b}^l(r)f(r)dr - \int_0^R K_{b,a}^l(r)g(r)dr - \sum_{n=0}^{N-1}\frac{a_n\psi_n(\nu_l)}{E_l}}{\sigma_l}\right\}^2 \\
&+ \left\langle\frac{1}{\sigma^2}\right\rangle\Lambda\int_0^R\left\{\frac{d^2f}{dr^2}\right\}^2 + \left\{\frac{d^2g}{dr^2}\right\}^2 dr
\end{aligned} \tag{18}$$

where $S_l$ is the frequency shifts to due changes in the stellar structure, $\sigma_l$ the corresponding error, $K^l_{a,b}$, $K^l_{b,a}$ the corresponding structural kernels, $\left\langle \frac{1}{\sigma^2} \right\rangle = \frac{1}{L}\left(\sum_{l=1}^{L} \frac{1}{\sigma_l^2}\right)$, and $\sum_{n=0}^{N-1} \frac{a_n \psi_n(\nu_l)}{E_l}$ an ad-hoc way of modelling surface effects (where $(\psi_n)_{n\in[0,N-1]}$ is a set of polynomials). The functions $f$ and $g$ and coefficients $a_n$ that minimise $J$ correspond to the inversion result. $\Lambda$ is a trade-off parameter between conforming to data and regularising the functions $f$ and $g$, and can be regulated in the `Structural inversion` tab. $N$ is the number of polynomials used to model surface effects and can also be modified in the `Structural inversion` tab. An additional Lagrangian constraint can be added to maintain a constant mass for the star when one of the two structural profiles ($a$ or $b$) is the density profile.

### 3.2.5 SOLA

There are two separate minimisations, one for each structural profile:

$$\begin{aligned} J(c_l(r_0)) &= \int_0^R \left\{T(r_0,r) - K_{\text{avg.}}(r_0,r)\right\}^2 dr + \beta \int_0^R \left\{K_{\text{cross}}(r_0,r)\right\}^2 w(r) dr \\ &+ \frac{\tan\theta \sum_{l=1}^{L} \left(c_l(r_0)\sigma_l\right)^2}{\langle\sigma^2\rangle} + \lambda\left\{1 - \int_0^R K_{\text{avg.}}(r_0,r) dr\right\} \\ &+ \sum_{n=0}^{N-1} a_n \sum_{l=1}^{L} \frac{c_l(r_0)\psi_n(\nu_l)}{E_l} \end{aligned} \tag{19}$$

$$\begin{aligned} J'(c'_l(r_0)) &= \beta' \int_0^R \left\{K'_{\text{cross}}(r_0,r)\right\}^2 w(r) dr + \int_0^R \left\{T'(r_0,r) - K'_{\text{avg.}}(r_0,r)\right\}^2 dr \\ &+ \frac{\tan\theta' \sum_{l=1}^{L} \left(c'_l(r_0)\sigma_l\right)^2}{\langle\sigma^2\rangle} + \lambda'\left\{1 - \int_0^R K'_{\text{avg.}}(r_0,r) dr\right\} \\ &+ \sum_{n=0}^{N-1} a'_n \sum_{l=1}^{L} \frac{c'_l(r_0)\psi_n(\nu_l)}{E_l} \end{aligned} \tag{20}$$

where

$$\begin{aligned} K_{\text{avg.}}(r_0,r) &= \sum_{l=1}^{L} c_l(r_0) K^l_{a,b}(r), \qquad K_{\text{cross}}(r_0,r) = \sum_{l=1}^{L} c_l(r_0) K^l_{b,a}(r), \\ K'_{\text{avg.}}(r_0,r) &= \sum_{l=1}^{L} c'_l(r_0) K^l_{b,a}(r), \qquad K'_{\text{cross}}(r_0,r) = \sum_{l=1}^{L} c'_l(r_0) K^l_{a,b}(r), \\ w(r) &= (1+r)^4 \\ &= \text{weight function used to suppress near-surface structure in } K_{\text{cross}} \text{ and } K'_{\text{cross}} \end{aligned}$$

The target functions $T(r_0,r)$ and $T'(r_0,r)$ are defined according to Eq. (5), each with their own separate width parameters, $\Delta$ and $\Delta'$. The parameters $\beta$ and $\beta'$ regulate the balance between optimising the averaging kernels and reducing the cross-term kernels. The parameters $\theta$ and $\theta'$ regulate the balance between optimising the averaging kernel and reducing the 1-sigma error. The parameters $\beta$, $\theta$, $\Delta$ and $\beta'$, $\theta'$, $\Delta'$ can be adjusted independently in the `Structural inversion` tab – the choice of structural function (given by the structural functions check boxes) determines which set of 3 parameters can be adjusted. The coefficients $a_n$ and $a'_n$ are Lagrangian multipliers used to cancel the unknown

surface contributions. The solutions are then constructed as follows:

$$f(r_0) = \sum_{l=1}^{L} c_l(r_0) S_l \tag{21}$$

$$g(r_0) = \sum_{l=1}^{L} c'_l(r_0) S_l \tag{22}$$

Keeping the stellar mass constant can be used as an extra constraint provided one of the two structural profiles ($a$ or $b$) is the the density profile.

### 3.2.6 Various error bars and kernels

As was done for the rotational inversions, it is possible to define averaging kernels, and vertical/horizontal error bars. However, since the inversion involves two unknown functions, another kernel called the cross-term kernel also intervenes. To see this, one can start by expressing the first inverted structural function as follows:

$$\frac{\delta a_{\rm inv.}(r_0)}{a(r_0)} = \sum_{l=1}^{L} c_l(r_0) S_l \tag{23}$$

If inserted into Eq. (10), this yields:

$$\begin{aligned} \frac{\delta a_{\rm inv.}(r_0)}{a(r_0)} &= \int_0^R \sum_{l=1}^{L} c_l(r_0) K^l_{a,b}(r) \frac{\delta a(r)}{a(r)} dr + \int_0^R \sum_{l=1}^{L} c_l(r_0) K^l_{b,a}(r) \frac{\delta b(r)}{b(r)} dr \\ &= \int_0^R K^l_{\rm avg.}(r) \frac{\delta a(r)}{a(r)} dr + \int_0^R K^l_{\rm cross}(r) \frac{\delta b(r)}{b(r)} dr \end{aligned}$$

where we have neglected surface contributions. From this, one can see that $K_{\rm cross}$ gives an idea of the amount of cross-talk coming from the second, unknown structural function $\delta b/b$. An analogous cross-term kernel can also be defined when inverting for the second structural kernel.

## 3.3 Mean density estimates

The mean density estimates are done in the $\underline{\rho}$ `inversion` tab using 3 different methods: a SOLA inversion, a scaling law based on the average large frequency separation $\langle \Delta\nu \rangle$, and a scaling law based on the surface correcting procedure described in Kjeldsen et al. (2008, ApJ 683, L175). These three methods can be put in the following form:

$$\underline{\rho}_{\rm inv} = \underline{\rho}_{\rm ref} s^2 \tag{24}$$

where

$$\underline{\rho} = \frac{M}{V} = \frac{3M}{4\pi R^3} \tag{25}$$

$$s = \frac{1}{2} \sum_{l=1}^{L} c_l \frac{\nu_l^{\rm obs}}{\nu_l^{\rm ref}} = 1 + \frac{1}{2} \sum_{\ell=1}^{L} c_l \frac{\delta\nu_l}{\nu_l} = 1 + \frac{1}{2} \frac{\delta \underline{\rho}_{\rm inv}}{\underline{\rho}} \tag{26}$$

This form represents a non-linear extension to a linear inversion or a linearised scaling law, the $c_l$ being the corresponding linear coefficients. The quantity $s$ is the scale factor

by which the reference model needs to be scaled so that linear inversion theory yields no further correction to the mean density. Using this approach has the advantage of fully retrieving the scaling laws (with a slight modification to Kjeldsen et al's approach) while allowing the use of linear inversion coefficients which can then be used to construct averaging and cross-term kernels for the 3 methods (see Reese et al. 2011, A&A, submitted). In what follows, we describe how the $c_l$ coefficients are obtained for each of the 3 methods.

### 3.3.1 SOLA

The $c_l$ coefficients are obtained by minimising the following cost function:

$$\begin{aligned} J(c_l) &= \int_0^1 \left\{K_{\rm avg}(x) - T(x)\right\}^2 dx + \beta \int_0^1 \left\{K_{\rm cross}(x)\right\}^2 dx + \frac{\tan\theta \sum_{l=1}^{L} \left(c_l \sigma_l\right)^2}{\langle \sigma^2 \rangle} \\ &+ \lambda \left\{ 1 - \frac{1}{2} \sum_{l=1}^{L} c_l \right\} + \sum_{n=0}^{N-1} a_n \sum_{l=1}^{L} \frac{c_l \psi_n(\nu_l)}{E_l} \end{aligned} \tag{27}$$

where $x = r/R$ and where $K_{\rm avg}$, $K_{\rm cross}$ and $T$ are defined in Eqs. (35)-(37). Here, the constraint $1 = \frac{1}{2}\sum_{l=1}^{L} c_l$, imposed using the Lagrangian multiplier $\lambda$, ensures that the inversion result is exact for a homologous modification of the model.

### 3.3.2 $\langle \Delta\nu \rangle$ scaling law

The $c_l$ coefficients are deduced from the following relation:

$$\frac{\delta \underline{\rho}_{\rm inv}}{\underline{\rho}} = 2 \frac{\delta \langle \Delta\nu \rangle}{\langle \Delta\nu \rangle} = \sum_{l=1}^{L} c_l \frac{\delta\nu_l}{\nu_l} \tag{28}$$

where:

$$\langle \Delta\nu \rangle = \frac{\sum_\ell \left(N_\ell - 1\right) \langle \Delta\nu(\ell) \rangle}{\sum_\ell N_\ell - 1} \tag{29}$$

$$\langle \Delta\nu(\ell) \rangle = \frac{\sum_{i=1}^{N_\ell} \left[\nu_i(\ell) - \langle \nu(\ell) \rangle\right] \left[n_i(\ell) - \langle n(\ell) \rangle\right]}{\sum_{i=1}^{N_\ell} \left[n_i(\ell) - \langle n(\ell) \rangle\right]^2}, \tag{30}$$

$$\langle \nu(\ell) \rangle = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \nu_i(\ell), \tag{31}$$

$$\langle n(\ell) \rangle = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} n_i(\ell). \tag{32}$$

The above definition uses a weighted average of the least-square estimates of the mean large frequency separation from Kjeldsen et al. (2008) for each $\ell$ value. Using a least-squares estimate has the advantage of not requiring consecutive radial orders to estimate the large frequency separation.

### 3.3.3 Kjeldsen et al's approach

The $c_l$ coefficients are deduced from the following relation:

$$\frac{\delta \underline{\rho}_{\rm inv}}{\underline{\rho}} = 2 \frac{b \frac{\delta\langle\nu\rangle}{\langle\nu\rangle} - \frac{\delta\langle\Delta\nu\rangle}{\langle\Delta\nu\rangle}}{b - 1} = \sum_{l=1}^{L} c_l \frac{\delta\nu_l}{\nu_l} \tag{33}$$

where

$$\langle \nu \rangle = \frac{\sum_\ell (N_\ell - 1) \langle \nu(\ell) \rangle}{\sum_\ell N_\ell - 1}, \tag{34}$$

and where Eqs. (29)-(32) continue to apply. Equation (34) is an average of the frequency, such that the weighting corresponds to what is used in Eq. (29).

### 3.3.4 Various error bars and kernels

As was done for the structural inversions, we define averaging and cross-term kernels, and a target function, $T$, as follows:

$$K_{\text{avg.}}(x) = \sum_{l=1}^{L} c_l K^l_{\rho,b}(x) \tag{35}$$

$$K_{\text{cross}}(x) = \sum_{l=1}^{L} c_l K^l_{b,\rho}(x) \tag{36}$$

$$T(x) = \frac{4\pi\rho x^2 R^3}{M} \tag{37}$$

These then enter the following expression for the error, which takes into account the fact that the reference model is scaled by $s$:

$$\frac{\underline{\rho}_{\text{inv}} - \underline{\rho}_{\text{obs}}}{s^2 \underline{\rho}_{\text{ref}}} = \int_0^1 (K_{\text{avg}}(x) - T(x)) \frac{\rho_{\text{obs}} - s^2 \rho_{\text{ref}}}{s^2 \rho_{\text{ref}}} dx + \int_0^1 K_{\text{cross}}(x) \frac{b_{\text{obs}} - s^k b_{\text{ref}}}{s^k b_{\text{ref}}} dx \tag{38}$$

where we have neglected surface terms, errors on the frequency shifts and other sources of error such as non-linear effects not accounted for in Eq. (10). The value of the exponent $k$ depends on which structural profile is represented by $b$. If $b = \Gamma_1$, then $k = 0$, whereas if $b = c^2$, then $k = 2$ (the second case is based on the assumption that the reference model has the same radius as the observed star). An upper bound on the error (neglecting surface terms and other sources of error) can be obtained through the Cauchy-Schwartz inequality:

$$\left| \frac{\underline{\rho}_{\text{inv}} - \underline{\rho}_{\text{obs}}}{s^2 \underline{\rho}_{\text{ref}}} \right| \leq \|K_{\text{avg}} - T\|_2 \left\| \frac{\rho_{\text{obs}} - s^2 \rho_{\text{ref}}}{s^2 \rho_{\text{ref}}} \right\|_2 + \|K_{\text{cross}}\|_2 \left\| \frac{b_{\text{obs}} - s^k b_{\text{ref}}}{s^k b_{\text{ref}}} \right\|_2 \tag{39}$$

where $\|f\|_2 = \sqrt{\int_0^1 f^2}$. The 1-$\sigma$ error bar around the inverted density, based on the measurement errors on the relative frequency shifts, is given by the following formula, which takes into account the non-linear extension:

$$\sigma_{\underline{\rho}_{\text{inv}}} = \underline{\rho}_{\text{ref}} s \sqrt{\sum_{l=1}^{L} c_l^2 \sigma_l^2} \tag{40}$$

### 3.3.5 Displayed quantities

Various quantities which appear in Eqs. (38) and (39) are displayed in the $\underline{\rho}$ `inversion` tab. Here are some explanations:

- $\underline{\rho}_{\text{ref}}$: density of reference model from which the inversion and scaling relations are applied.
- $\underline{\rho}_{\text{obs}}$: density of the observed "star". This is deduced from target structural profiles if they have been loaded. These are assumed to represent the true difference between the reference model and the observed star.
- $\underline{\rho}_{\text{inv}} = s^2 \underline{\rho}_{\text{ref}}$: density estimate from inversion.
- $\sigma_{\underline{\rho}}$, as based on Eq. (40).
- $\delta\underline{\rho}/\underline{\rho} = \left(\underline{\rho}_{\text{inv}} - \underline{\rho}_{\text{ref}}\right)/\underline{\rho}_{\text{ref}} = s^2 - 1$: Relative variation on the mean density, deduced from inversion.
- $\|\Delta K_{\text{avg}}\|_2 = \sqrt{\int_0^1 \left(K_{\text{avg}}(x) - T(x)\right)^2 dx}$.
- $\|K_{\text{cross}}\|_2 = \sqrt{\int_0^1 K_{\text{cross}}^2(x) dx}$.
- $\int \Delta K_{\text{avg}} \delta\rho/\rho = \int_0^1 \left(K_{\text{avg}}(x) - T(x)\right) \frac{\rho_{\text{obs}}(x) - s^2\rho_{\text{ref}}(x)}{s^2\rho_{\text{ref}}(x)} dx$. True error from mismatch on averaging kernel. This formula takes into account the fact that the reference model is scaled by $s$.
- $\int K_{\text{cross}} \delta b/b = \int_0^1 K_{\text{cross}}(x) \frac{b_{\text{obs}}(x) - s^k b_{\text{ref}}(x)}{s^k b_{\text{ref}}(x)} dx$. True error from cross-talk. This formula takes into account the fact that the reference model is scaled by $s$.
- $\text{Surf}[i] = \sum_{l=1}^{L} \frac{c_l \psi_i(\nu_l)}{E_l}$. This is the $i^{\text{th}}$ surface terms. The number of terms displayed in the window can be adjusted by modifying the parameter `n_surf_disp` in `MeanDensityEstimate.java` and recompiling `InversionKit`.

These quantities as well as the inversion parameters can be saved in a text file using the `Save results` button.

## 3.4 Integration method

`InversionKit` also gives a choice of the integration method which is used in the different inversions. These choices have different effects depending on whether an RLS or a SOLA inversion is applied. The two options are "Sliding window" and "Chebyshev". Their effects are as follows:

- **RLS**
  - *Sliding window*: The kernels are kept on the original grid and the output inverted function(s) expressed on a smaller output grid where the points are distributed according to the inverse of the sound velocity (*i.e.*, there are more grid points where the sound velocity is smaller). Accordingly, integrations are carried out on the original (large) grid, using weights which are based on an interpolation within a sliding window of the function to be integrated (this is somewhat analogous to what is done when using finite differences to calculate derivatives). The inverted function is expressed on a b-spline basis deduced from the output grid and the regularisation matrix is based on the analytical derivatives of the b-splines. The RLS procedure then searches for optimal

coefficients over the b-spline basis before calculating the values of the inverted function(s) on the output grid.

  - *Chebyshev*: The kernels and the output inverted function(s) are expressed on the Gauss collocation grid associated with Chebyshev polynomials. The integrals and the regularisation matrix are all based on this grid, the coefficients being deduced from a spectral approach. The RLS procedure searches directly for the optimal function values on this (output) grid.

- **SOLA**

  - *Sliding window*: The kernels are kept on the original grid and the output inverted function(s) expressed on a smaller output grid where the points are distributed according to the inverse of the sound velocity. Accordingly, integrations are carried out on the original grid, using weights which are based on an interpolation within a sliding window of the function to be integrated.
  - *Chebyshev*: The kernels and the output inverted function(s) are expressed on the Gauss collocation grid associated with Chebyshev polynomials. The integrals are carried out on this grid using coefficients based on a spectral approach.

*Remark*: The "Sliding window" approach is slower than the "Chebyshev" approach but yields results which are more accurate and which do not depend on the number of grid points used in the output grid (since the integration grid does not depend on the output grid, as opposed to the "Chebyshev" approach).

# 4 Known bugs

Here is a list of known bugs. If you find any other, please let us know by sending us an email (Daniel.Reese@obspm.fr).

- excessive zooming on plots can produce irregular behaviour

# 5 Copyright notices

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

```
Copyright (c) Daniel Reese, Sergei Zharkov, 2008, 2009, 2011

This file is part of InversionKit.

InversionKit 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.

InversionKit 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 InversionKit.  If not, see <http://www.gnu.org/licenses/>.
```

## 5.1 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 `InversionKit`.

## 5.2 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 MIDROSYSTEMS, 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.